MSX 2

Zakboekje

Wessel Akkermans



Wessel Akkermans

Zakboekje

MSX2
Zakboekje

MSX 2
Zakboekje

Wessel Akkermans

uitgeverij STARK-TEXEL
postbus 302 - 1794 ZG Oosterend tel. 02223 - 661

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

MSX

MSX2. –Oosterend : Stark-Texel
Zakboekje / Wessel Akkermans.
ISBN 90 6398 224 0
SISO 365.3 UDC 681.3 NUGI 851
Trefw.: MSX2 (computer).

1e druk 1987
ISBN 90 6398 224 0

Voorwoord

Het MSX-zakboekje, dat werd uitgegeven in 1985, was alleen bedoeld voor MSX1-computers. Inmiddels heeft de MSX2-computer zijn intrede gedaan. Er ontstond een grote vraag naar een nieuw zakboekje, waarin ook MSX2-gegevens zouden zijn opgenomen. Vandaar dit nieuwe zakboekje. In dit boekje heb ik getracht zoveel mogelijk gegevens over alle MSX-computers samen te brengen.

Nu zijn er niet alleen nieuwe MSX2-gegevens in dit boek aan de oude MSX1-informatie toegevoegd, er zijn ook bestaande MSX1-gegevens toegevoegd aan de oude informatie en andere gegevens zijn ge-update. Het gevolg is dan ook, dat de totale omvang van het boek bijna is verdubbeld.

Als u zelf, net als ik, al enige tijd bezig bent met het programmeren van uw MSX-computer, dan zult u hebben ontdekt, dat u tijdens het programmeren vaak met meerdere boeken en een aantal tijdschriften tegelijk aan het werk moet zijn. Dit is niet echt handig. Daarom heb ik alle gegevens, die ik in de laatste 2 jaar wel eens heb gebruikt, kort samengevat in dit ene boek.

Dit houdt in, dat het hier gaat om een naslagwerk en niet om een studie boek. Ik heb dan ook meer mijn best gedaan om de informatie zo kort mogelijk samen te vatten, dan om er uitgebreide uitleg bij te geven. Toch hoop ik dat de gegeven uitleg voldoende is voor iedereen die de basisbeginselen van het programmeren onder de knie heeft.

Voor de moeilijkste onderwerpen heb ik wat meer uitleg toegevoegd. Bovendien heb ik een aantal programma's toegevoegd aan het einde van het boek. Al die programma's hebben tot doel bepaalde moeilijk uit te leggen principes aan de hand van een praktijkvoorbeeld duidelijk te maken. Een aantal van die programma's is bovendien tegelijkertijd direct toepasbaar als utility.

Om dit naslagwerk goed toegankelijk te maken heb ik de onderwerpen zoveel mogelijk samengevoegd in achtereenvolgens de volgende clusters:

- Algemene (conversie) tabellen.
- BASIC-tabellen.
- Machinetaal-tabellen.
- Interface-connectors.

- BIOS-entry points, BDOS-calls en systeemlocaties.
- Geluidsprocessor en video-processor.
- Voorbeeldprogramma's.

Tenslotte is een trefwoordenlijst opgenomen, die u in staat stelt de gewenste gegevens zeer snel terug te vinden.

februari 1987
Wessel Akkermans

Inhoud

1 Conversies: decimaal, binair, octaal en hexadecimaal

dec.	binair	oct.	hex.	dec.	binair	oct.	hex.
0	0000 0000	000	00	42	0010 1010	052	2A
1	0000 0001	001	01	43	0010 1011	053	2B
2	0000 0010	002	02	44	0010 1100	054	2C
3	0000 0011	003	03	45	0010 1101	055	2D
4	0000 0100	004	04	46	0010 1110	056	2E
5	0000 0101	005	05	47	0010 1111	057	2F
6	0000 0110	006	06	48	0011 0000	060	30
7	0000 0111	007	07	49	0011 0001	061	31
8	0000 1000	010	08	50	0011 0010	062	32
9	0000 1001	011	09	51	0011 0011	063	33
10	0000 1010	012	0A	52	0011 0100	064	34
11	0000 1011	013	0B	53	0011 0101	065	35
12	0000 1100	014	0C	54	0011 0110	066	36
13	0000 1101	015	0D	55	0011 0111	067	37
14	0000 1110	016	0E	56	0011 1000	070	38
15	0000 1111	017	0F	57	0011 1001	071	39
16	0001 0000	020	10	58	0011 1010	072	3A
17	0001 0001	021	11	59	0011 1011	073	3B
18	0001 0010	022	12	60	0011 1100	074	3C
19	0001 0011	023	13	61	0011 1101	075	3D
20	0001 0100	024	14	62	0011 1110	076	3E
21	0001 0101	025	15	63	0011 1111	077	3F
22	0001 0110	026	16	64	0100 0000	100	40
23	0001 0111	027	17	65	0100 0001	101	41
24	0001 1000	030	18	66	0100 0010	102	42
25	0001 1001	031	19	67	0100 0011	103	43
26	0001 1010	032	1A	68	0100 0100	104	44
27	0001 1011	033	1B	69	0100 0101	105	45
28	0001 1100	034	1C	70	0100 0110	106	46
29	0001 1101	035	1D	71	0100 0111	107	47
30	0001 1110	036	1E	72	0100 1000	110	48
31	0001 1111	037	1F	73	0100 1001	111	49
32	0010 0000	040	20	74	0100 1010	112	4A
33	0010 0001	041	21	75	0100 1011	113	4B
34	0010 0010	042	22	76	0100 1100	114	4C
35	0010 0011	043	23	77	0100 1101	115	4D
36	0010 0100	044	24	78	0100 1110	116	4E
37	0010 0101	045	25	79	0100 1111	117	4F
38	0010 0110	046	26	80	0101 0000	120	50
39	0010 0111	047	27	81	0101 0001	121	51
40	0010 1000	050	28	82	0101 0010	122	52
41	0010 1001	051	29	83	0101 0011	123	53

dec.	binair	oct.	hex.	dec.	binair	oct.	hex.
84	0101 0100	124	54	130	1000 0010	202	82
85	0101 0101	125	55	131	1000 0011	203	83
86	0101 0110	126	56	132	1000 0100	204	84
87	0101 0111	127	57	133	1000 0101	205	85
88	0101 1000	130	58	134	1000 0110	206	86
89	0101 1001	131	59	135	1000 0111	207	87
90	0101 1010	132	5A	136	1000 1000	210	88
91	0101 1011	133	5B	137	1000 1001	211	89
92	0101 1100	134	5C	138	1000 1010	212	8A
93	0101 1101	135	5D	139	1000 1011	213	8B
94	0101 1110	136	5E	140	1000 1100	214	8C
95	0101 1111	137	5F	141	1000 1101	215	8D
96	0110 0000	140	60	142	1000 1110	216	8E
97	0110 0001	141	61	143	1000 1111	217	8F
98	0110 0010	142	62	144	1001 0000	220	90
99	0110 0011	143	63	145	1001 0001	221	91
100	0110 0100	144	64	146	1001 0010	222	92
101	0110 0101	145	65	147	1001 0011	223	93
102	0110 0110	146	66	148	1001 0100	224	94
103	0110 0111	147	67	149	1001 0101	225	95
104	0110 1000	150	68	150	1001 0110	226	96
105	0110 1001	151	69	151	1001 0111	227	97
106	0110 1010	152	6A	152	1001 1000	230	98
107	0110 1011	153	6B	153	1001 1001	231	99
108	0110 1100	154	6C	154	1001 1010	232	9A
109	0110 1101	155	6D	155	1001 1011	233	9B
110	0110 1110	156	6E	156	1001 1100	234	9C
111	0110 1111	157	6F	157	1001 1101	235	9D
112	0111 0000	160	70	158	1001 1110	236	9E
113	0111 0001	161	71	159	1001 1111	237	9F
114	0111 0010	162	72	160	1010 0000	240	A0
115	0111 0011	163	73	161	1010 0001	241	A1
116	0111 0100	164	74	162	1010 0010	242	A2
117	0111 0101	165	75	163	1010 0011	243	A3
118	0111 0110	166	76	164	1010 0100	244	A4
119	0111 0111	167	77	165	1010 0101	245	A5
120	0111 1000	170	78	166	1010 0110	246	A6
121	0111 1001	171	79	167	1010 0111	247	A7
122	0111 1010	172	7A	168	1010 1000	250	A8
123	0111 1011	173	7B	169	1010 1001	251	A9
124	0111 1100	174	7C	170	1010 1010	252	AA
125	0111 1101	175	7D	171	1010 1011	253	AB
126	0111 1110	176	7E	172	1010 1100	254	AC
127	0111 1111	177	7F	173	1010 1101	255	AD
128	1000 0000	200	80	174	1010 1110	256	AE
129	1000 0001	201	81	175	1010 1111	257	AF

dec.	binair	oct.	hex.	dec.	binair	oct.	hex.
176	1011 0000	260	B0	222	1101 1110	336	DE
177	1011 0001	261	B1	223	1101 1111	337	DF
178	1011 0010	262	B2	224	1110 0000	340	E0
179	1011 0011	263	B3	225	1110 0001	341	E1
180	1011 0100	264	B4	226	1110 0010	342	E2
181	1011 0101	265	B5	227	1110 0011	343	E3
182	1011 0110	266	B6	228	1110 0100	344	E4
183	1011 0111	267	B7	229	1110 0101	345	E5
184	1011 1000	270	B8	230	1110 0110	346	E6
185	1011 1001	271	B9	231	1110 0111	347	E7
186	1011 1010	272	BA	232	1110 1000	350	E8
187	1011 1011	273	BB	233	1110 1001	351	E9
188	1011 1100	274	BC	234	1110 1010	352	EA
189	1011 1101	275	BD	235	1110 1011	353	EB
190	1011 1110	276	BE	236	1110 1100	354	EC
191	1011 1111	277	BF	237	1110 1101	355	ED
192	1100 0000	300	C0	238	1110 1110	356	EE
193	1100 0001	301	C1	239	1110 1111	357	EF
194	1100 0010	302	C2	240	1111 0000	360	F0
195	1100 0011	303	C3	241	1111 0001	361	F1
196	1100 0100	304	C4	242	1111 0010	362	F2
197	1100 0101	305	C5	243	1111 0011	363	F3
198	1100 0110	306	C6	244	1111 0100	364	F4
199	1100 0111	307	C7	245	1111 0101	365	F5
200	1100 1000	310	C8	246	1111 0110	366	F6
201	1100 1001	311	C9	247	1111 0111	367	F7
202	1100 1010	312	CA	248	1111 1000	370	F8
203	1100 1011	313	CB	249	1111 1001	371	F9
204	1100 1100	314	CC	250	1111 1010	372	FA
205	1100 1101	315	CD	251	1111 1011	373	FB
206	1100 1110	316	CE	252	1111 1100	374	FC
207	1100 1111	317	CF	253	1111 1101	375	FD
208	1101 0000	320	D0	254	1111 1110	376	FE
209	1101 0001	321	D1	255	1111 1111	377	FF
210	1101 0010	322	D2				
211	1101 0011	323	D3				
212	1101 0100	324	D4				
213	1101 0101	325	D5				
214	1101 0110	326	D6				
215	1101 0111	327	D7				
216	1101 1000	330	D8				
217	1101 1001	331	D9				
218	1101 1010	332	DA				
219	1101 1011	333	DB				
220	1101 1100	334	DC				
221	1101 1101	335	DD				

2 Two's complement tabel voor negatieve waarden

rechter tetrade	linker tetrade							
	8	9	A	B	C	D	E	F
0	-128	-112	-96	-80	-64	-48	-32	-16
1	-127	-111	-95	-79	-63	-47	-31	-15
2	-126	-110	-94	-78	-62	-46	-30	-14
3	-125	-109	-93	-77	-61	-45	-29	-13
4	-124	-108	-92	-76	-60	-44	-28	-12
5	-123	-107	-91	-75	-59	-43	-27	-11
6	-122	-106	-90	-74	-58	-42	-26	-10
7	-121	-105	-89	-73	-57	-41	-25	-9
8	-120	-104	-88	-72	-56	-40	-24	-8
9	-119	-103	-87	-71	-55	-39	-23	-7
A	-118	-102	-86	-70	-54	-38	-22	-6
B	-117	-101	-85	-69	-53	-37	-21	-5
C	-116	-100	-84	-68	-52	-36	-20	-4
D	-115	-99	-83	-67	-51	-35	-19	-3
E	-114	-98	-82	-66	-50	-34	-18	-2
F	-113	-97	-81	-65	-49	-33	-17	-1

3 Machten van 2, 8 en 16

```
Machten van 2:

2^0  =          1
2^1  =          2
2^2  =          4
2^3  =          8
2^4  =         16
2^5  =         32
2^6  =         64
2^7  =        128
2^8  =        256
2^9  =        512
2^10 =       1024
2^11 =       2048
2^12 =       4096
2^13 =       8192
2^14 =      16384
2^15 =      32768
2^16 =      65536
2^17 =     131072
2^18 =     262144
2^19 =     524288
2^20 =    1048576
2^21 =    2097152
2^22 =    4194304
2^23 =    8388608
2^24 =   16777216
2^25 =   33554432
2^26 =   67108864
2^27 =  134217728
2^28 =  268435456
2^29 =  536870912
2^30 = 1073741824
2^31 = 2147483648
2^32 = 4294967296
```

```
Machten van 8:

8^0  =          1
8^1  =          8
8^2  =         64
8^3  =        512
8^4  =       4096
8^5  =      32768
8^6  =     262144
8^7  =    2097152
8^8  =   16777216
8^9  =  134217728
8^10 = 1073741824
```

```
Machten van 16:

16^0 =          1
16^1 =         16
16^2 =        256
16^3 =       4096
16^4 =      65536
16^5 =    1048576
16^6 =   16777216
16^7 =  268435456
16^8 = 4294967296
```

4 Afgeleide goniometrische functies

Afgeleide functie	MSX-notatie
Secans	1/COS(X)
Cosecans	1/SIN(X)
Cotangens	1/TAN(X)
Inverse sinus	ATN(X/SQR(-X*X+1))
Inverse cosinus	-ATN(X/SQR(-X*X+1))+1.5708
Inverse secans	ATN(X/SQR(X*X-1))+SGN(SGN(X)-1) *1.5708
Inverse cosecans	ATN(X/SQR(X*X-1))+(SGN(X)-1) *1.5708
Inverse cotangens	ATN(X)+1.5708
Hyperbolische sinus	(EXP(X)-EXP(-X))/2
Hyperbolische cosinus	(EXP(X)+EXP(-X))/2
Hyperbolische tangens	EXP(-X)/EXP(X)+EXP(-X)*2+1
Hyperbolische secans	2/(EXP(X)+EXP(-X))
Hyperbolische cosecans	2/(EXP(X)-EXP(-X))
Hyperbolische cotangens	EXP(-X)/(EXP(X)-EXP(-X))*2+1
Inv. hyperb. sinus	LOG(X+SQR(X*X+1))
Inv. hyperb. cosinus	LOG(X+SQR(X*X-1))
Inv. hyperb. tangens	LOG((1+X)/(1-X))/2
Inv. hyperb. secans	LOG((SQR(-X*X+1)+1)/X)
Inv. hyperb. cosecans	LOG((SGN(X)*SQR(X*X+1)+1)/X)
Inv. hyperb. cotangens	LOG((X+1)/(X-1))/2

5 Operators

Functionele operators

Prioriteit	Operator	Betekenis
1	functies	Wanneer in expressies functies worden gebruikt, dan zullen deze met de hoogste prioriteit worden verwerkt.

Rekenkundige operators

Prioriteit	Operator	Betekenis
2	^	Machtsverheffen
3	*	vermenigvuldigen
	/	delen
	\	delen van gehele getallen
	MOD	rest bepaling
4	+	optellen
	-	aftrekken

Vergelijkende operators

Prioriteit	Operator	Betekenis
5	=	Is gelijk aan
	<	Is kleiner dan
	>	Is groter dan
	<> of ><	Is ongelijk aan
	<= of =<	Is kleiner dan of gelijk aan
	>= of =>	Is groter dan of gelijk aan

Logische operators

Prioriteit	Operator	Betekenis
6	AND	EN
	OR	OF
	NOT	NIET
	XOR	Exclusieve OF
	EQV	Gelijkwaardig met
	IMP	Impliceert dat

6 Kleurnummers

Kleurnummer	Kleur
0	transparant
1	zwart
2	groen
3	licht groen
4	donker blauw
5	licht blauw
6	donker rood
7	blauw
8	rood
9	licht rood
10	donker geel
11	licht geel
12	donker groen
13	steen rood
14	grijs
15	wit

7 Variabelen en waardetoekenning

Soorten variabelen:

soort	soortbepaling	waardetoekenning
Alfanumeriek	$ achter naam	LET N$="ABC"
Integer numeriek	% achter naam	LET A%=5 De toe te kennen waarde mag -32768 tot +32767 zijn, en moet een geheel getal zijn.
Numeriek met enkelvoudige nauwkeurigheid	! achter naam	LET A!=123456 De toe te kennen waarde mag uit 6 cijfers bestaan. Daarboven wordt het getal afgerond.
Numeriek met dubbele nauwkeurigheid	# achter naam	LET A#=12345678901 23 De toe te kennen waarde mag uit 14 cijfers bestaan. Daarboven wordt het getal afgerond.

Indien een variabelenaam niet wordt gevolgd door $, %, ! of #, dan zal die variabele als een numerieke variabele met dubbele nauwkeurigheid worden beschouwd.

Soorten constanten:

soort	omschrijving
Integer	Waarde tussen -32768 en +32767.
Fixed point met enkele nauwkeurigheid	Waarde wordt gevolgd door het ! teken. Voorbeeld: 12.3!

(vervolg soorten constanten)

soort	omschrijving
Fixed point met dubbele nauwkeurigheid	Waarde wordt gevolgd door het # teken. Voorbeeld: 12.3#
Floating point met enkele nauwkeurigheid	Waarde wordt in wetenschappelijke notatie geschreven, waarbij E het exponentteken is. Voorbeeld: 12.3E4
Floating point met dubbele nauwkeurigheid	Waarde wordt in wetenschappelijke notatie geschreven, waarbij D het exponentteken is. Voorbeeld: 12.3D9
Hexadecimaal	Waarde wordt voorafgegaan door &H. Voorbeeld: &HFF (= decimaal 255).
Octaal	Waarde wordt voorafgegaan door &O. Voorbeeld: &O377 (= decimaal 255).
Binair	Waarde wordt voorafgegaan door &B. Voorbeeld: &B11111111 (= decimaal 255).

8 MSX-BASIC instructieset

In dit hoofdstuk zullen alle BASIC-statements, -functies en -commando's op alfabetische volgorde worden behandeld. Daarbij zullen statements, die alleen voor MSX2 gelden, een aparte aanduiding (MSX2) krijgen.

<> Wat tussen gehoekte haakjes staat vormt 1 item.
[] Wat tussen vierkante haken staat mag worden weggelaten.
() Normale haakjes maken moeten worden ingetikt op de plaats, waar ze in de syntax-beschrijving staan.
:;., Deze leestekens moeten worden ingetikt op de plaats, waar ze in de syntax staan.

Items, die in hoofdletters staan, moeten worden ingetikt, zoals ze in de syntax-beschrijving staan.

In de hierna volgende opsomming wordt van ieder statement, van iedere functie en van ieder commando, achtereenvolgens de syntax, het type en een omschrijving gegeven. Om het geheel zo compact mogelijk te houden, is afgezien van voorbeelden. Alleen daar, waar een voorbeeld de omschrijving korter kan maken, is een voorbeeld gebruikt.

Syntax: ABS(numerieke uitdrukking)
Soort: Functie.
Omschrijving: Geeft de absolute waarde van de numerieke uitdrukking. (Bijv.: ABS(-2)=2).

Syntax: ASC(alfanumerieke uitdrukking)
Soort: Functie.
Omschrijving: Geeft de ASCII-code van het eerste teken uit de alfanumerieke uitdrukking. (Bijv.: ASC("abc")=97).

Syntax: ATN(numerieke uitdrukking)
Soort: Functie.
Omschrijving: Geeft de arctangens van de goniometrische verhouding uit de numerieke uitdrukking in radialen.

Syntax: AUTO[<regelnummer>[,[<stapgrootte>]]]
Soort: Commando.
Omschrijving: Start automatisch regelnummeren.
Indien stapgrootte wordt weggelaten, dan zal nummeren starten bij regelnummer en zal de laatst gebruikte stapgrootte worden aangehouden.
Indien de komma en stapgroote worden weggelaten, zal vanaf regelnummer worden genummerd met een stapgrootte van 10.
Indien alle parameters worden weggelaten zal worden genummerd vanaf regelnummer 10 met een stapgrootte van 10.
CONTROL+STOP-toets = Beeindigen van automatische regelnummering.
* achter een regelnummer betekent dat de betreffende regel al in het geheugen staat.

Syntax: BASE(numerieke uitdrukking)
Soort: Systeemvariabele met MSX2 uitbreiding.
Omschrijving: Vertegenwoordigt het startadres in video RAM van de tabel die met de numerieke uitdrukking wordt aangegeven.
De startadressen van de tabellen 0 tot en met 19 mogen worden gewijzigd.
Voor MSX1 computers mag de numerieke uitdrukking een waarde hebben van 0 tot en met 19.
Voor MSX2 computers mag de numerieke uitdrukking een waarde hebben van 0 tot en met 44.
De numerieke expressie heeft de volgende betekenis:

0	Scherm-info tabel	(SCREEN 0)
2	Matrix-tabel	(SCREEN 0)
5	Scherm-info tabel	(SCREEN 1)
6	kleurtabel	(SCREEN 1)
7	Matrix-tabel	(SCREEN 1)
8	Sprite-info tabel	(SCREEN 1)
9	SPRITE$-tabel	(SCREEN 1)
10	Scherm-info tabel	(SCREEN 2)
11	kleurtabel	(SCREEN 2)
12	Matrix-tabel	(SCREEN 2)
13	Sprite-info tabel	(SCREEN 2)

14	SPRITE$-tabel	(SCREEN 2)
15	Scherm-info tabel	(SCREEN 3)
17	Matrix-tabel	(SCREEN 3)
18	Sprite-info tabel	(SCREEN 3)
19	SPRITE$-tabel	(SCREEN 3)
20	Scherm-info tabel	(SCREEN 4)
21	kleurtabel	(SCREEN 4)
22	Matrix-tabel	(SCREEN 4)
23	Sprite-info tabel	(SCREEN 4)
24	SPRITE$-tabel	(SCREEN 4)
25	Scherm-info tabel	(SCREEN 5)
26	kleurtabel	(SCREEN 5)
27	Matrix-tabel	(SCREEN 5)
28	Sprite-info tabel	(SCREEN 5)
29	SPRITE$-tabel	(SCREEN 5)
30	Scherm-info tabel	(SCREEN 6)
31	kleurtabel	(SCREEN 6)
32	Matrix-tabel	(SCREEN 6)
33	Sprite-info tabel	(SCREEN 6)
34	SPRITE$-tabel	(SCREEN 6)
35	Scherm-info tabel	(SCREEN 7)
36	kleurtabel	(SCREEN 7)
37	Matrix-tabel	(SCREEN 7)
38	Sprite-info tabel	(SCREEN 7)
39	SPRITE$-tabel	(SCREEN 7)
40	Scherm-info tabel	(SCREEN 8)
41	kleurtabel	(SCREEN 8)
42	Matrix-tabel	(SCREEN 8)
43	Sprite-info tabel	(SCREEN 8)
44	SPRITE$-tabel	(SCREEN 8)

Syntax: BEEP
Soort: Statement.
Omschrijving: Veroorzaakt een pieptoon.

Syntax: BIN$(numerieke uitdrukking)
Soort: Functie.
Omschrijving: Geeft de binaire weergave van de waarde in de numerieke uitdrukking. De numerieke uitdrukking mag in waarde varieren van -32768 tot +32767. Indien de waarde negatief is, zal het resultaat in two's complement worden weergegeven.

Syntax: BLOAD "<dev1>:<file-naam>"[,R[,<displ>]]
BLOAD "<dev2>:<file-naam>",S

Soort: Statement met MSX2 uitbreiding.

Omschrijving: Laadt het opgegeven bestand (machinetaal programma of scherminhoud) in het geheugen.

dev1 = CAS, A, B, C, D, E, F.
dev2 = A, B, C, D, E, F.
R = Automatisch starten na laden.
displ = verplaatsing binnen het geheugen ten opzichte van de geheugenplaats tijdens BSAVE. Indien geen displ wordt opgegeven zal deze 0 zijn.
S = Scherminhoud.
Voor SCREEN-modes 4 t/m 7 (MSX2) geldt, dat de actieve pagina wordt geladen.

Syntax: BSAVE "<dev1>:<file-naam>",<b>,<e>[,<s>]
BSAVE "<dev2>:<file-naam>",<b>,<e>,S

Soort: Statement met MSX2 uitbreiding.

Omschrijving: Schrijft het geheugengebied (RAM of video-RAM) vanaf adres <b> tot en met adres <e> naar het aangegeven bestand.

dev1 = CAS, A, B, C, D, E, F.
dev2 = A, B, C, D, E, F.
<b> = beginadres van geheugengebied.
<e> = eindadres van geheugengebied.
<s> = startadres van machinetaalprogr.
S = Scherminhoud.
Voor SCREEN-modes 4 t/m 7 (MSX2) geldt, dat alleen de actieve pagina wordt opgeslagen.

Syntax: CALL <exp. statement> [(parameters)]

Soort: Statement.

Omschrijving: Roept een statement in een ROM-cartridge aan. Eventuele parameter(s) worden doorgegeven aan het betreffende statement.
Hierna volgt een opsomming van ROM-statements, die met CALL kunnen worden aangeroepen. Deze CALL's werken alleen, wanneer de betreffende ROMs op de computer zijn aangesloten.

CALL COM ([<channel>:],GOSUB <regelnummer>)

Bepaalt het regelnummer, waar naartoe moet worden gesprongen indien op het gegeven kanaal iets wordt ontvangen.

<channel> = kanaal nummer (zie ook OPEN-statement) mag varieren van 0 tot 2. Default is 0.

CALL COMBREAK(["<n>:"],<aantal>)

Verstuurt het opgegeven aantal breaktekens over kanaalnummer n. Dat aantal mag varieren van 3 tot 32767.

CALL COMDTR(["<n>:"],<numerieke uitdrukking>)

Indien de numerieke uitdrukking 0 is, wordt het signaal DTR ge-reset. Voor alle andere waarden van de numerieke uitdrukking wordt DTR ge-set.

CALL COMHELP[(<n>:)]

Hiermee wordt een lijst afgedrukt van de parameters, volgens welke het aangegeven kanaal (n) is geinitialiseerd. De lijst ziet er als volgt uit:

CALL COMINI ("
<device #>
<datalengte>
<pariteit>
<stopbitlengte>
<Xon/Xoff>
<CTS-handshake>
<auto LF bij ontvangst CR>
<auto LF na zenden CR>
<SI/SO>"
,<Rxbd>
,<Txbd>
,<time>)

De default lijst ziet er als volgt uit:

CALL COMINI("0:8N1XHNNN",1200,1200,0)

```
CALL COMINI [(<str>][,[<Rxbd>][,[<Txbd>][,[<time>]]]])]
               Initialiseert  het RS232-kanaal  met  de
               gegeven parameters.
               <str>  = "[n:][l[p[b[x[h[i[s[c]]]]]]]]"
                          |   | | | | | | | |
Kanaalnummer -------------    | | | | | | | |
     (van 0 - 2)              | | | | | | | |
Lengte van de code -----------  | | | | | | |
     (5, 6, 7 of 8 bits)        | | | | | | |
Parity Flag --------------------  | | | | | |
     (E (even), O (odd),          | | | | | |
      I (ignore), N (no parity)   | | | | | |
Stopbit lengte -------------------  | | | | |
     (1 =1, 2 =1-1/2, 3 =2          | | | | |
Xon/Xoff control -------------------  | | | |
     (X =enable, N =disable)          | | | |
CTS/RTS handshake --------------------  | | |
     (H =handshake, N =geen handshake)  | | |
Insert LF after CR is received ---------  | |
     (A =insert, N =not insert)           | |
Send LF after CR -------------------------  |
     (A =niet zenden, N =zenden)            |
Shift-in/out control (DC1/3) ---------------
     (S =enable, N=disable)

               <Rxbd> = Baudrate voor ontvangst
               <Txbd> = Baudrate voor zenden
                        Mogelijke Baudrates:
                        50, 75, 110, 300, 600, 1200,
                        1800, 2000, 2400, 3600, 4800,
                        7200, 9600, 19200.
                        Indien  voor  Txbd geen  waarde
                        wordt  opgegeven,  zal deze  de
                        waarde van Rxbd krijgen.
               <time> = Het aantal seconden waarna  een
                        time-out  error moet worden ge-
                        genereerd.
                        Indien 0 wordt  opgegeven,  zal
                        er  geen time-out error  worden
                        gegenereerd.
```

CALL COMON("[<n>:]")

Na uitvoering van dit statement zal het systeem steeds, voordat een volgend statement wordt uitgevoerd, controleren of er op kanaal n een teken is ontvangen. Is dit het geval, dan zal het programma worden onderbroken en zal de subroutine, die in het statement CALL COM(n:),GOSUB is aangegeven, worden uitgevoerd.

CALL COMOFF("[<n>:]")

Na uitvoering van dit statement zal het systeem niet meer kijken of er op kanaal n iets is ontvangen.

CALL COMSTAT(["<n>:"],<numerieke variabele naam>)

De status van kanaal n wordt in de aangegeven variabele opgeslagen, waarbij de bitwaarde de volgende betekenis heeft indien het bit is ge-set:

bit 15 - buffer overflow.
bit 14 - time out error.
bit 13 - framing error.
bit 12 - overrun error.
bit 11 - parity error.
bit 10 - control/break key ingedrukt.
bit 9 - gereserveerd.
bit 8 - gereserveerd.
bit 7 - Clear To Send.
bit 6 - Timer output-2 asserted.
bit 5 - gereserveerd.
bit 4 - gereserveerd.
bit 3 - Data Set Ready.
bit 2 - Break detected.
bit 1 - Ring indicator aan.
bit 0 - Carrier detected.

CALL COMSTOP("[<n>:]")

Na uitvoering van dit statement zal het systeem nog wel kijken of er op kanaal n een teken is ontvangen, doch het lopende programma wordt daarvoor niet onderbroken. Zodra echter een statement CALL COMON is gegeven, zal het lopende programma alsnog worden onderbroken, indien er tenminste een teken was ontvangen.

CALL COMTERM[("<n>:")]

Start de terminal emulatie mode voor kanaal n. De functietoetsen krijgen hierbij de volgende functies:

F6: Literal mode aan/uit.
In deze mode worden controlcodes weergegeven als de code + &H40.

F7: Half/Full duplex mode.
In Half Duplex mode worden tekens die naar kanaal n worden gestuurd ook teruggezonden naar het beeldscherm.

F8: Printer-echo aan/uit.
Indien printer-echo aan is, worden alle tekens die naar het scherm worden gestuurd ook naar de printer gestuurd.

CALL FORMAT

Formateerd een floppy disk. Na het geven van dit statement vraagt het systeem welke floppy moet worden geformateerd. Hierop kan het betreffende drive-nummer (A, B, C, D, E of F) worden ingegeven.

CALL MEMINI[(<aantal bytes>)]

Reserveert een deel van het geheugen (met de in "aantal bytes" opgegeven grootte) voor gebruik als RAM-disk. De opgegeven grootte moet een geheel getal zijn, dat groter is dan 1023, of dat 0 is. De werkelijke grootte van het RAM-disk gebied zal altijd een veelvoud van 256 bytes zijn.
De RAM-disk is op dezelfde manier toe-

gankelijk als normale disks, alleen is de device-indicatie van RAM-disk MEM. (Voorbeeld: SAVE "MEM:fnaam")

CALL MFILES

Geeft een lijst van de files, die op de RAM-disk staan, en de vrije ruimte, op het beeldscherm weer.

CALL MKILL("filenaam")

Verwijdert het bestand met de naam "filenaam" van de RAM-disk.

CALL MNAME("filenaam1" AS "filenaam2")

Vervangt de naam van "filenaam1" door "filenaam2". Het te hernoemen bestand dient op RAM-disk te staan.

CALL SYSTEM

Verlaat BASIC en keert terug naar MSXDOS. Dit commando werkt alleen indien BASIC was opgestart vanuit MSXDOS.

Syntax:	CDBL(numerieke uitdrukking)
Soort:	Functie.
Omschrijving:	Levert een getal in dubbele nauwkeurigheid.

Syntax:	CHR$(numerieke uitdrukking)
Soort:	Functie.
Omschrijving:	Levert het teken op, waarvan de code in de numerieke uitdrukking staat.

Syntax:	CINT(numerieke uitdrukking)
Soort:	Functie.
Omschrijving:	Converteert de waarde uit de numerieke uitdrukking naar een integer-getal. De waarde mag varieren van -32768 tot 32767.

Syntax: CIRCLE[STEP](x,y),r,[k],[bh],[eh],[af]

Soort: Statement met MSX2 uitbreiding.

Omschrijving: Tekent een cirkelboog met als middelpunt de coordinaten x en y, met een straal r, in de kleur k, vanaf beginhoek bh tot eindhoek eh, met een afplatting volgens de afplattingsfactor af.

x - mag voor SCREEN 2, 3, 4, 5 en 8 een geheel getal van 0 tot 255 zijn.
mag voor SCREEN 6 en 7 een geheel getal van 0 tot 511 zijn.

y - mag voor SCREEN 2, 3 en 4 een geheel getal van 0 tot 191 zijn.
mag voor SCREEN 5, 6, 7 en 8 een geheel getal van 0 tot 211 zijn.

r - mag eventueel zo groot zijn dat de cirkel buiten het beeld valt.

k - voor SCREEN 2, 3, 4, 5 en 7 een geheel getal van 0 tot 15.
voor SCREEN 6 een geheel getal van 0 tot 3.
voor SCREEN 8 een geheel getal van 0 tot 255.

bh - de hoek vanaf waar de cirkel zal worden getekend, opgegeven in radialen (tussen 0 en 2pi). Default 0.
Door de hoek als negatieve waarde op te geven, zal het begin van de cirkelboog worden verbonden met het middelpunt.

eh - de hoek tot waar de cirkel zal worden getrokken, opgegeven in radialen (tussen 0 en 2pi). Default 2pi.
Door de hoek als negatieve waarde op te geven, zal het einde van de cirkelboog worden verbonden met het middelpunt.

af - De verhouding in afstand tussen horizontale en tussen verticale pixels. Default 1.

Syntax: CLEAR [stringmem[,highmem]]
Soort: Statement.
Omschrijving: Bepaalt de grootte van het geheugengebied, waarin de inhoud van stringvariabelen wordt opgeborgen op het aantal bytes in stringmem en maakt een stuk geheugen vanaf adres highmem tot het door het systeem gebruikte RAM vrij voor machinecode.
Als neveneffect van het CLEAR-statement zullen nog geopende bestanden worden gesloten, gedimensioneerde arrays worden gewist en voorgeprogrammeerde functies zoals DEF FN, DEFINT, etc. worden geëlimineerd.

Syntax: CLOAD [?]["prognaam"]
Soort: Statement.
Omschrijving: Laadt het aangegeven programma van cassette in het geheugen. Indien geen programmanaam wordt opgegeven wordt het eerstvolgende programma van cassette gelezen.
Indien de optie ? wordt gebruikt, heeft dit statement een Verify-functie. Het aangegeven programma wordt dan wel van cassette gelezen, doch niet in het geheugen geschreven, maar vergeleken met het al in het geheugen staande programma.

Syntax: CLOSE [[#]b1[,[#]b2].....[,[#]bn]]
Soort: Statement.
Omschrijving: Sluit de bestanden b1 tot en met bn af. b1 tot en met bn zijn gehele getallen, die varieren van 1 tot 15. Indien geen parameters aan het CLOSE-statement worden toegevoegd, zullen alle bestanden worden gesloten.

Syntax: CLS
Soort: Statement.
Omschrijving: Maakt het beeldscherm schoon en plaatst de cursor in de linker bovenhoek.

Syntax: COLOR [vk][,ak][,rk]

Soort: Statement.

Omschrijving: Kent voorgrond-, achtergrond- en randkleuren toe.

Met de numerieke uitdrukkingen vk (voorgrond kleur), ak (achtergrond kleur) en rk (rand kleur) wordt een keuze uit het kleurenpalet gemaakt. Alle kleuren zijn samengesteld uit bepaalde intensiteiten van de kleuren rood groen en blauw. De default samenstellingen zijn als volgt:

palet-	intensiteit			
nummer	rood	groen	blauw	kleur
0	0	0	0	transparant
1	0	0	0	zwart
2	1	6	1	groen
3	3	7	3	lichtgroen
4	1	1	7	donkerblauw
5	2	3	7	blauw
6	5	1	1	donkerrood
7	2	6	7	lichtblauw
8	7	1	1	rood
9	7	3	3	lichtrood
10	6	6	1	donkergeel
11	6	6	4	lichtgeel
12	1	4	1	donkergroen
13	6	2	5	paars
14	5	5	5	grijs
15	7	7	7	wit

MSX2-uitbreidingen:
De default waarden van het kleurenpalet kunnen worden gewijzigd met het COLOR=-statement.

Voor SCREEN 6 geldt, dat voor vk, ak en rk slechts een waarde van 0 tot 3 mag worden opgegeven. Voor de randkleur geldt bovendien het volgende:

rk = 0 tot 3 --> de hiervoor gegeven paletkleur wordt gebruikt.

rk = 16 tot 31 --> wordt gedecodeerd in twee paletnummers, die elk een waarde van 0 tot 3 hebben. Het ene nummer is

voor even pixels het andere voor oneven pixels. Op die manier zijn verticale strepen te verkrijgen. Decodering vindt als volgt plaats:

rk	paletkleur even pixels	paletkleur oneven pixels
16	0	0
17	0	1
18	0	2
19	0	3
20	1	0
21	1	1
22	1	2
23	1	3
24	2	0
25	2	1
26	2	2
27	2	3
28	3	0
29	3	1
30	3	2
31	3	3

Het volgende programma laat enkele mogelijkheden in SCREEN 6 zien:

```
10 SCREEN 6
20 COLOR=(1,7,0,0)
30 COLOR=(2,0,7,0)
40 COLOR=(3,0,0,7)
50 FOR I=16 TO 31
60 COLOR ,,I
70 FOR J=1 TO 500:NEXT J
80 NEXT I
90 GOTO 90
```

Voor SCREEN 8 geldt, dat voor vk, ak en rk een waarde van 0 tot 255 kan worden opgegeven. In deze mode zijn vk, ak en rk geen paletnummers, doch geven zij direct de kleurintensiteiten voor rood, groen en blauw, en wel als volgt:

Kleur = 4*rood + 32*groen + blauw

rood en groen kunnen hierin een intensiteit van 0 tot 7 hebben en blauw van 0 tot 3.

Syntax: COLOR=(<paletnr>,<rood>,<groen>,<blauw>)
Soort: MSX2-statement.
Omschrijving: Kent een nieuwe combinatie van kleurintensiteiten toe aan het paletnummer.
<paletnr> = een geheel getal van 0-15.
<rood> = een geheel getal van 0-7.
<groen> = een geheel getal van 0-7.
<blauw> = een geheel getal van 0-7.
Dit statement werkt niet in SCREEN 8.
Met: VDP(9)=VDP(9)OR&H20 wordt palet nummer 0 niet-transparant gemaakt en kan er een normale kleur aan worden toegekend. Met VDP(9)=VDP(9)AND&HDF wordt paletnumer 0 weer transparant.

Syntax: COLOR[=NEW]
Soort: MSX2-statement.
Omschrijving: Zet de kleurintensiteiten terug op hun default waarden, zoals omschreven bij het statement COLOR.

Syntax: COLOR=RESTORE
Soort: MSX2-statement.
Omschrijving: Zet de kleurintensiteiten op de waarden die in het video-RAM zijn opgeslagen. Afhankelijk van de SCREEN-mode zijn de kleurintensiteiten op de volgende geheugenadressen opgeslagen (de lengte van de tabel is altijd &H20 bytes):

SCREEN 0 (40)	&H0400
SCREEN 0 (80)	&H0F00
SCREEN 1	&H2020
SCREEN 2	&H2020
SCREEN 3	&H2020
SCREEN 4	&H2020
SCREEN 5	&H7680+(&H8000*<pagenr>)
SCREEN 6	&H7680+(&H8000*<pagenr>)
SCREEN 7	&HFA80+(&H10000*<pagenr>)
SCREEN 8	&HFA80+(&H10000*<pagenr>)

Syntax: COLOR SPRITE(<spritenr>)=<waarde>
Soort: MSX2-statement.
Omschrijving: Geeft in SCREEN-modes 4, 5, 6, 7 en 8 het met <spritenr> aangegeven sprite een kleur volgens de opgegeven <waarde>. De <waarde> bestaat uit de volgende componenten:

- het paletnummer (0-15) plus:
- eventueel een waarde 32.
- eventueel een waarde 64.

De waarde 32 opgeteld bij het paletnummer wil zeggen: ON SPRITE GOSUB detecteert geen botsingen met deze sprite.
De waarde 64 opgeteld bij het paletnummer wil zeggen: ON SPRITE GOSUB detecteert geen botsingen met deze sprite, deze sprite wordt onzichtbaar. Echter, indien deze sprite een andere sprite, die niet van de waarde 64 is voorzien, overlapt, dan zal het overlappende deel wel zichtbaar worden. De kleuren van de beide elkaar overlappende sprite-delen zullen een logische OR ondergaan.

Syntax: COLOR SPRITE$(spritenr)=<string-uitdr.>
Soort: MSX2-statement.
Omschrijving: Werkt net zoals COLOR SPRITE(), doch nu wordt per lijn, waaruit het sprite is opgebouwd, een aparte kleur aangegeven. De codes uit de string-uitdrukking worden als volgt samengesteld:

128 verplaats sprite 32 posities naar links.
64 detecteer botsingen niet en laat elkaar overlappende sprite-delen een logische OR ondergaan.
32 detecteer botsingen niet.
0-15 paletnummer.

De op deze manier samengestelde codes kunnen bijvoorbeeld met CHR$(<code>) in het statement worden gezet.

Syntax: CONT
Soort: Commando
Omschrijving: Vervolgt het programma met de regel, waarop het was onderbroken. Indien het programma werd onderbroken met het STOP-statement, dan zal CONT het programma voortzetten met de regel na het STOP-statement.

Syntax: COPY (<x1>,<y1>)-(<x2>,<y2>)[,<src>] TO (<x3>,<y3>)[,<dest>[,<logop>]]
Soort: MSX2-statement.
Omschrijving: Copieert van video-RAM naar video-RAM. Het blok dat wordt gecopieerd, is de rechthoek, waarvan de coordinaten x1,y1 en x2,y2 de diagonaal vormen. De coordinaten x3,y3 geven de linker bovenhoek aan van het blok waar naartoe wordt gecopieerd.
Werkt alleen voor SCREEN 5, 6, 7 en 8.
Zie voor verdere specificatie van src, dest en logop de beschrijving na het laatste COPY-statement.

Syntax: COPY (<x1>,<y1>)-(<x2>,<y2>)[,<src>] TO <array>
Soort: MSX2-statement.
Omschrijving: Copieert van video-RAM naar een array. Het blok dat wordt gecopieerd, is de rechthoek, waarvan de coordinaten x1,y1 en x2,y2 de diagonaal vormen. De array dient een numerieke array te zijn, die groot genoeg is om het te copieren beeldscherm gedeelte te bevatten. De grootte wordt bepaald met de formule:

```
INT((P*ABS(x2-x1)+1)*(ABS(y2-y1)+1)+7)/8+4
```

P = 4 in SCREEN-modes 5, 7 en 8.
P = 2 in SCREEN-mode 6.

Werkt alleen voor SCREEN 5, 6, 7 en 8.
Zie voor verdere specificatie van src en array de beschrijving na het laatste COPY-statement.

Syntax: COPY (<x1>,<y1>)-(<x2>,<y2>)[,<src>] TO "[<dev>:]<fnam>"

Soort: MSX2-statement.

Omschrijving: Copieert van video-RAM naar een bestand. Het blok dat wordt gecopieerd, is de rechthoek, waarvan de coordinaten x1,y1 en x2,y2 de diagonaal aangeven.
Werkt alleen voor SCREEN 5, 6, 7 en 8.
Zie voor verdere specificatie van src, dev en fnam de beschrijving na het laatste COPY-statement.

Syntax: COPY <array>[,<dir>] TO (<x3>,<y3>)[,<dest>[,<logop>]]

Soort: MSX2-statement.

Omschrijving: Copieert van array naar video-RAM.
De inhoud van de array wordt naar een blok in het video-RAM geschreven, zodanig, dat de linker bovenhoek van de rechthoek op het scherm de coordinaten x3,y3 heeft. Het uitlezen van de array geschiedt in de met dir aangegeven richting. Bij het in het video-RAM schrijven van de data kan een logische bewerking worden uitgevoerd, zoals aangegeven in logop.
Werkt alleen voor SCREEN 5, 6, 7 en 8.
Zie voor een verdere specificatie van array, dir, dest en logop de beschrijving na het laatste COPY-statement.

Syntax: COPY "[<dev>:]<fnam>"[,<dir>] TO (<x3>,<y3>)[,<dest>[,<logop>]]

Soort: MSX2-statement.

Omschrijving: Copieert van bestand naar video-RAM.
De inhoud van het bestand <fnam> op schijf <dev> wordt naar video-RAM geschreven, zodanig dat de linkerbovenhoek van de rechthoek op het scherm de coordinaten x3,y3 heeft. Het uitlezen van het bestand geschiedt in de met dir aangegeven richting. Bij het in het video-RAM schrijven van de data kan een logische bewerking worden uitgevoerd, zoals aangegeven in logop.

Werkt alleen voor SCREEN 5, 6, 7 en 8.
Zie voor een verdere specificatie van dev, fnam, dir, dest en logop de beschrijving na het laatste COPY-statement.

Syntax: COPY <array> TO "[<dev>:]<fnam>"
Soort: MSX2-statement.
Omschrijving: Copieert een numerieke array, waarin een gedeelte van het grafisch scherm is opgeslagen, naar een bestand op schijf.
Werkt alleen voor SCREEN 5, 6, 7 en 8.
Zie voor een verdere specificatie van array, dev en fnam de beschrijving na het laatste COPY-statement.

Syntax: COPY "[<dev>:]<fnam>" TO <array>
Soort: MSX2-statement.
Omschrijving: Copieert een bestand op schijf, waarin een gedeelte van het grafisch scherm is opgeslagen, naar een numerieke array.
Werkt alleen voor SCREEN 5, 6, 7 en 8.
Zie voor een verdere specificatie van dev, fnam en array de beschrijving na het laatste COPY-statement.

Syntax: COPY "[<dev>:]<fnam>"TO"[<dev>:][<fnam>]
Soort: Statement.
Omschrijving: Copieert een diskette-bestand.
Indien het bron-device wordt weggelaten, zal de default of laatst gebruikte schijf worden gelezen. Indien het bestemmings-device wordt weggelaten, dan zal dezelfde schijf worden gebruikt om naar toe te schrijven als het brondevice.
Indien de naam van het bestemmingsbestand wordt weggelaten, zal de bestandsnaam van het bronbestand worden gebruikt.
In de bestandsnaam van het bronbestand mogen asteriksjes en vraagtekens worden gebruikt ter vervanging van letters. Op die manier kunnen met 1 statement meerdere bestanden tegelijk worden gecopieerd.

Beschrijving van de in de COPY-statements gebruikte items:

array= naam van een gedimensioneerde numerieke variabele.
dest = doel page-nummer:
(0-3 voor SCREEN 5 en 6)
(0-1 voor SCREEN 7 en 8)
dev = apparaat aanduiding:
(A, B, C, D, E, F)
(+ MEM voor copieren van apparaat naar apparaat.)
dir = richting (0-3):
0 = linksboven naar rechtsonder.
1 = rechtsboven naar linksonder.
2 = linksonder naar rechtsboven.
3 = rechtsonder naar linksboven.
fnam = file-naam (8 tekens, eventueel gevolgd door een punt en 3 tekens).
logop= Logische bewerking:
AND, OR, PSET, PRESET, XOR,
TAND, TOR, TPSET, TPRESET, TXOR.
Indien <logop> wordt weggelaten, dan wordt PSET uitgevoerd.
De logische bewerkingen worden uitgevoerd op de binaire waarden van de kleurnummers van de bron- en doel-pixels. De letter T voor de logische bewerking wil zeggen dat transparante kleuren geen effect hebben.
src = bron page-nummer:
(0-3 voor SCREEN 5 en 6)
(0-1 voor SCREEN 7 en 8)

Syntax: COPY SCREEN [<mode>]
Soort: MSX2-statement.
Omschrijving: Digitaliseert een van buiten komend video-beeld.
Indien mode=0, dan wordt het gedigitaliseerde beeld opgeslagen in de active pagina van het video-RAM.
Indien mode=1, dan worden twee beelden gedigitaliseerd en opgeslagen in de ac-

tive pagina van het video-RAM minus 1 en in de active pagina zelf. In dit geval dient de active pagina een oneven nummer te hebben. De twee beelden vormen samen 1 frame.
De default waarde van mode is 0.
Indien de mode wordt weggelaten, wordt de laatst gebruikte mode gekozen.
Werkt alleen op MSX2-computers waarop een video-digitizer is aangesloten.

Syntax: COS(<radialen>)
Soort: Functie.
Omschrijving: Berekent de cosinus van het aangegeven aantal radialen.

Syntax: CSAVE "<prognaam>"[,<snelheid>]
Soort: Statement.
Omschrijving: Schrijft het programma in het geheugen naar cassette onder de in <prognaam> gegeven naam. De programmanaam mag slechts 6 tekens lang zijn. De default schrijfsnelheid is 1200 baud. Deze kan worden gewijzigd, door op de plaats van <snelheid> een 1 of een 2 te zetten.
<snelheid> = 1 - 1200 baud.
<snelheid> = 2 - 2400 baud.

Syntax: CSNG(<numerieke uitdrukking>)
Soort: Functie.
Omschrijving: Converteert de waarde uit de numerieke uitdrukking naar een getal met enkele precisie.

Syntax: CSRLIN
Soort: Functie.
Omschrijving: Geeft, in SCREEN 0 en 1, het regelnummer waarop de cursor staat.

Syntax: CVD(<string-variabele>)
Soort: Functie.
Omschrijving: Converteert de alfanumerieke variabele naar een acht bytes dubbele precisie waarde. Werkt alleen voor disk-systemen.

Syntax: CVI(<string-variabele>)
Soort: Functie.
Omschrijving: Converteert de alfanumerieke variabele naar een twee bytes integer waarde. Werkt alleen voor disk-systemen.

Syntax: CVS(<string-variabele>)
Soort: Functie.
Omschrijving: Converteert de alfanumerieke variabele naar een vier bytes enkele precisie waarde. Werkt alleen voor disk-systemen.

Syntax: DATA <const>[,<const>.....]
Soort: Statement.
Omschrijving: Het DATA-statement mag numerieke- en alfanumerieke constanten door elkaar bevatten. De constanten dienen van elkaar te zijn gescheiden door een komma.
De constanten worden met een READ-statement gelezen.
Alfanumerieke constanten mogen eventueel tussen aanhalingstekens worden geplaatst. Dit is nodig voor bijvoorbeeld voorlopende spaties.

Syntax: DEF FN<naam>[(<var>[,<var>..])=<formule>
Soort: Statement.
Omschrijving: Definieert een door de programmeur samengestelde functie.
naam = De naam, die we aan de functie geven, en waaronder we later de functie kunnen oproepen. Indien de functienaam wordt gevolgd door een dollarteken ($), dan dient de formule betrekking te hebben op strings.
var = een variabele, die in de formule voorkomt en daarin moet worden ingevuld.
formule= een numerieke of alfanumerieke uitdrukking.

Syntax: DEFDBL<letter>[,<letter>...]
DEFDBL<letter>-<letter>
Soort: Statement.
Omschrijving: Maakt alle variabelen die met de aangegeven letters beginnen van het type dubbele precisie.

Syntax: DEFINT<letter>[,<letter>...]
DEFINT<letter>-<letter>
Soort: Statement.
Omschrijving: Maakt alle variabelen, die met de aangegeven letters beginnen, van het type integer.

Syntax: DEFSNG<letter>[,<letter>...]
DEFSTR<letter>-<letter>
Soort: Statement.
Omschrijving: Maakt alle variabelen, die met de aangegeven letters beginnen, van het type enkele nauwkeurigheid.

Syntax: DEFSTR<letter>[,<letter>...]
DEFSTR<letter>-<letter>
Soort: Statement.
Omschrijving: Maakt van alle variabelen, die met de aangegeven letters beginnen, alfanumerieke variabelen.

Syntax: DEFUSR[<nummer>]=<startadres>
Soort: Statement.
Omschrijving: Hiermee wordt aan een machinetaalprogramma een volgnummer gegeven en wordt het startadres daarvan vastgelegd.
Indien het nummer wordt weggelaten, dan wordt nummer 0 genomen.

Syntax: DELETE [<regelnr>][-<regelnr>]
Soort: Commando.
Omschrijving: Verwijdert alle aangegeven regelnummers uit het BASIC-programma.
Indien het eerste regelnummer wordt weggelaten, zullen alle regels tot en met het laatst aangegeven regelnummer worden verwijderd.
Indien het laatste regelnummer wordt

weggelaten, zal alleen het aangegeven regelnummer worden verwijderd.

Syntax: DIM<var>(<dim1>[,<dim2>]...)

Soort: Statement.

Omschrijving: Creeert ruimte in het geheugen voor een array van <dim1> elementen. Met <dim2> kan het aantal twee dimensionale elementen worden aangegeven. Met <dim3> het aantal drie dimensionale elementen, etc.

<var> = naam van numerieke of alfanumerieke variabele.

<dim> = aantal elementen in een dimensie.

Een array kan pas opnieuw worden gedimensioneerd, nadat de inhoud is gewist met het ERASE-statement.
Indien een variabele niet wordt gedimensioneerd, bevat deze variabele 11 elementen (0-10).

Syntax: DRAW <string>

Soort: Macro-statement met MSX2 uitbreiding.

Omschrijving: Tekent een lijn in de grafische modes, volgens de in de string gegeven aanwijzingen. In deze string kunnen de volgende subcommando's voorkomen:

[B][N]U<X> - X pixels omhoog.
[B][N]D<X> - X pixels omlaag.
[B][N]L<X> - X pixels naar links.
[B][N]R<X> - X pixels naar rechts.
[B][N]E<X> - X pixels rechtsomhoog.
[B][N]F<X> - X pixels rechtsomlaag.
[B][N]G<X> - X pixels linksomlaag.
[B][N]H<X> - X pixels linksomhoog.
[B][N]M<X>,<Y> - naar positie X,Y. Indien X en Y door het minteken worden voorafgegaan zal de verplaatsing relatief t.o.v. huidige positie zijn.

Hierin is:

[B] = Blank (trekt geen lijn)

[N] = Na trekken van de lijn wordt teruggekeerd naar de oorsprong.

Overige subcommando's:

C<kleur>	- Voor SCREEN 2, 3, 4, 5 en 7 een geheel getal van 0 - 15. Voor SCREEN 6 een geheel getal van 0 - 3. Voor SCREEN 8 een geheel getal van 0 - 255.
S<getal>	- Teken in schaal <getal>/4.
A<hoek>	- Teken onder aangegeven hoek. <hoek> = 0 t/m 3 voor 0, 90, 180 en 270 graden.
X<var>;	- Voer de string uit de aangegeven variabele uit.

In de subcommando's mogen in plaats van numerieke waarden ook numerieke variabelen worden gebruikt. In dat geval moet de variabele worden voorafgegaan door het is gelijk teken (=) en gevolgd door een puntkomma (;).

Voorbeeld:
```
LET A=50
DRAW "U=A;"
```

Syntax: DSKF(<disk>)
Soort: Functie.
Omschrijving: Geeft de vrije ruimte op de aangegeven schijf.
<disk> - 0 = default drive.
1 = drive A
2 = drive B
etc. (t/m 6)

Syntax: DSKI$(<disk>,<sector>)
Soort: Functie.
Omschrijving: Leest de aangegeven sector van de aangegeven schijf in het systeembuffer.
Het systeembuffer start op geheugenadres PEEK(62289)+256*PEEK(62290).

Syntax: DSKO$(<disk>,<sector>)
Soort: Statement.
Omschrijving: Schrijft het systeembuffer naar de aangegeven sector op de aangegeven schijf.
Het systeembuffer start op geheugenadres PEEK(62289)+256*PEEK(62290).
Door op deze adressen het startadres van

een variabele te zetten kan de inhoud van een variabele naar een sector worden geschreven (zie ook VARPTR).

Syntax: END
Soort: Statement.
Omschrijving: Beeindigt de uitvoering van een programma en sluit alle geopende bestanden.

Syntax: EOF(<filenummer>)
Soort: Functie.
Omschrijving: Indien het einde van de aangegeven file is bereikt, levert deze functie de waarde -1 op, zoniet, dan zal het resultaat 0 zijn.

Syntax: ERASE <var>[[,<var>]...]
Soort: Statement.
Omschrijving: Verwijdert de met DIM gecreeerde arrays uit het geheugen. Dit statement moet worden gebruikt, voordat een array opnieuw kan worden gedimensioneerd.

Syntax: ERL
Soort: Systeemvariabele.
Omschrijving: Bevat het regelnummer van de BASIC-regel waarin een fout is ontdekt.

Syntax: ERR
Soort: Systeemvariabele.
Omschrijving: Bevat de code van de fout, die tijdens het uitvoeren van een BASIC-regel is ontdekt.

Syntax: ERROR <foutcode>
Soort: Statement.
Omschrijving: Definieert eigen foutcodes. De foutcode kan een waarde van 0 tot 255 hebben. Indien een foutcode wordt gekozen, waarvoor geen foutboodschap aanwezig is, zal de tekst "unprintable error" worden afgedrukt.

Syntax: EXP(<macht>)
Soort: Functie.
Omschrijving: Geeft de waarde van e tot de aangegeven macht. De maximale waarde van <macht> mag 145.06286058562 zijn.

Syntax: FIELD[#]<fnum>,<chars>AS<fnam> [,<chars>AS<fnam>...]
Soort: Statement.
Omschrijving: Definieert de buffer lay-out voor random disk files. FIELDS moet worden gebruikt, nadat de file is geopend en voordat de statements GET en PUT worden gebruikt.
<fnum> = Het nummer waaronder de file is geopend.
<chars>= Het aantal tekens.
<fnam> = Naam van het veld (alfanumerieke variabele naam).
Het totaal aantal tekens mag niet meer dan 256 zijn.

Syntax: FILES ["[<dev>:][<fnam>]"]
Soort: Commando.
Omschrijving: Geeft een lijst van de op de aangegeven schijf staande bestanden op het beeldscherm.
Indien <dev> wordt weggelaten, wordt de default drive gelezen.
<fnam> mag ook vraagtekens en asterisken bevatten, zodat een bepaalde selectie van files kan worden gemaakt.

Syntax: FIX(<waarde>)
Soort: Functie.
Omschrijving: Geeft het gehele deel van de opgegeven waarde.

Syntax: FN<naam>[(<var>[,<var>]...)]
Soort: Functie.
Omschrijving: Roept de met <naam> aangegeven functie aan en geeft de eventueel meegegeven waarden (in <var>) door aan de aangeroepen functie. De functie moet met DEF FN zijn gedefinieerd.

Syntax: FOR <var>=<beginw>TO<eindw>[STEP<stapw>]
Soort: Statement.
Omschrijving: Geeft het begin van een programmalus aan, die eindigt bij het eerstvolgende NEXT-statement zonder variabelenaam of bij het eerstvolgende NEXT-statement met de in <var> gegeven variabele naam.
<var> is een numerieke variabele.
<beginw> dient lager dan <eindw> te zijn, indien geen stapwaarde of een positieve stapwaarde wordt opgegeven.
<eindw> dient lager dan <beginw> te zijn, indien een negatieve stapwaarde wordt opgegeven.
Indien geen stapwaarde wordt opgegeven, zal de stapwaarde 1 zijn.

Syntax: FRE(0)
FRE("")
Soort: Functie.
Omschrijving: FRE(0) levert de grootte (in bytes) van het nog voor BASIC beschikbare deel van het geheugen.
FRE("") levert de grootte (in bytes) van het nog vrije deel van het voor de inhoud van stringvariabelen gereserveerde geheugen (zie CLEAR).

Syntax: GET[#]<fnum>[,<recnum>]
Soort: Statement.
Omschrijving: Leest het in <recnum> gegeven recordnummer uit het random bestand dat was geopend onder nummer <fnum>. Indien geen recordnummer wordt gegeven, zal het eerstvolgende record worden gelezen.

Syntax: GET DATE<var>$[,A]
Soort: MSX2-statement.
Omschrijving: Leest de datum uit de klok-chip en zet deze in de aangegeven string variabele. Het formaat van de datum is afhankelijk van het type/fabrikaat MSX2-computer, maar bevat om ieder geval de dag, maand en jaar van elkaar gescheiden door een scheidingsteken (MM/DD/JJ of DD/MM/JJ of

JJ/MM/DD).
Indien optie A wordt toegevoegd, wordt alleen de dag (DD) in de aangegeven string variabele gezet.

Syntax: GET TIME<var>$[,A]
Soort: MSX2-statement.
Omschrijving: Leest de tijd uit de klok-chip en zet deze in de aangegeven string variabele (UU=uren, MM=minuten en SS=seconden). De uren, minuten en seconden zijn van elkaar gescheiden door een dubbele punt. Indien de optie A wordt toegevoegd, dan worden alleen de uren en minuten in de aangegeven stringvariabele gezet.

Syntax: GOSUB<regelnummer>
Soort: Statement.
Omschrijving: Springt naar het aangegeven regelnummer, doch onthoudt wat het volgende regelnummer zou zijn gewesst. Zodra een RETURN-statement wordt uitgevoerd, zal naar dat onthouden regelnummer worden teruggekeerd.

Syntax: GOTO<regelnummer>
Soort: Statement.
Omschrijving: Springt naar het aangegeven regelnummer.

Syntax: HEX$(<integer>)
Soort: Functie.
Omschrijving: Levert de hexadecimale weergave van het gehele getal in <integer>.
<integer> mag varieren van -32768 tot +32767. Indien <integer> een negatieve waarde heeft, zal de two's complement weergave worden gegeven.

Syntax: IF <expr> THEN <stat> [ELSE <stat>]
IF <expr> GOTO <regel> [ELSE <stat>]
Soort: Statement.
Omschrijving: Indien de uitdrukking in <expr> waar is, zal de rest van het statement worden uitgevoerd.
<stat> wil zeggen dat hiervoor in de

plaats een statement kan worden gedacht. Indien ELSE wordt gebruikt, zal het statement dat hierachter staat worden uitgevoerd indien de uitdrukking in <expr> niet waar is.

Syntax: INKEY$
Soort: Functie.
Omschrijving: Levert het teken, dat op het moment van uitvoering op het toetsenbord wordt ingetikt, op. Indien geen toets wordt ingedrukt tijdens het uitvoeren van deze functie, zal het resultaat een "empty string" zijn.

Syntax: INP(<poortnummer>)
Soort: Functie.
Omschrijving: Leest een byte van de aangegeven poort. Gebruik van deze functie maakt, dat uw BASIC-programma mogelijkerwijs niet op alle MSX-computer kan worden gebruikt, daar de poortnummers van computer tot computer kunnen verschillen.

Syntax: INPUT ["<tekst>";]<var>[,<var>...]
Soort: Functie.
Omschrijving: Drukt de gegeven <tekst> af op het beeldscherm en wacht daarna op data van het toetsenbord. Alle ingetikte data zal in de variabele worden opgeslagen. Het einde van de ingetikte data kan worden aangegeven met een komma, waarna de data voor de volgende variabele wordt verwacht, en met de RETURN-toets, waarna het statement als geeindigd wordt beschouwd.
Indien er meer gegevens worden ingetikt dan er variabelen achter het statement staan, zal de boodschap "Extra ignored" worden gegeven.
Indien er gegevens voor een numerieke variabele worden gevraagd, moeten de gegevens ook numeriek zijn.
Indien spaties voor of achter de ingave worden geplaatst, zullen deze vervallen,

tenzij de ingave tussen aanhalingstekens wordt gegeven.

Syntax: INPUT[#]<fnum>,<var>[,<var>...]
Soort: Statement.
Omschrijving: Leest gegevens uit de sequentiele file die is geopend onder nummer <fnum>. De file moet zijn geopend voor INPUT.
Voor het overige gelden dezelfde regels als voor het vorige INPUT-statement.

Syntax: INPUT$(<aantal>[,[#]<fnum>])
Soort: Functie.
Omschrijving: Leest het in <aantal> gegeven aantal tekens van het toetsenbord, of, indien een filenummer in <fnum> is gegeven, uit de gegeven file. De ingelezen tekens verschijnen niet op het beeldscherm.

Syntax: INSTR([<start>,]<onderzoek$>,<zoek$>)
Soort: Functie.
Omschrijving: Indien <zoek$> niet vanaf positie <start> binnen <onderzoek$> voorkomt zal het resultaat van de functie 0 zijn.
Indien <start> niet is opgenomen, zal <onderzoek$> vanaf het begin worden onderzocht.

Syntax: INT(<waarde>)
Soort: Functie.
Omschrijving: Levert het gehele deel van <waarde> op.

Syntax: INTERVAL [ON][OFF][STOP]
Soort: Statement.
Omschrijving: Met ON INTERVAL=x GOSUB wordt een interval van bepaalde lengte ge-set.
Met INTERVAL ON wordt na het uitvoeren van iedere statement gecontroleerd of het interval reeds is verstreken, en zo ja, dan wordt naar de aangegeven subroutine gesprongen.
Met INTERVAL OFF wordt niet langer gecontroleerd.
Met INTERVAL STOP wordt nog wel gecontroleerd, doch wordt na het verstrijken

van het interval alleen een vlag gezet en niet gesprongen. Zodra INTERVAL ON wordt gezet en de vlag geeft aan dat een interval was verstreken, zal alsnog worden gesprongen naar de aangegeven subroutine.

Syntax: KEY<functietoetsnr>,<string>
Soort: Statement.
Omschrijving: Kent de waarde uit <string> toe aan de aangegeven functietoets.

Syntax: KEY LIST
Soort: Statement.
Omschrijving: Resulteert in een lijst van alle, aan de functietoetsen toegekende, waarden, beginnende bij functietoets 1.

Syntax: KEY [ON][OFF]
Soort: Commando.
Omschrijving: Na KEY OFF worden de inhouden van de functietoetsen niet langer op de 24-ste regel afgedrukt. Na KEY ON verschijnen de teksten weer.

Syntax: KEY(<functietoetsnr>)[ON][OFF][STOP]
Soort: Statement.
Omschrijving: Na KEY(nr)ON wordt na uitvoering van ieder statement gecontroleerd of de aangegeven functietoets is ingedrukt, en indien dit zo is, dan wordt naar de subroutine gesprongen die is aangegeven in het statement ON KEY GOSUB.
Na KEY(nr)OFF wordt niet meer gecontroleerd of een functietoets is ingedrukt geweest.
Na KEY(nr)STOP wordt wel gecontroleerd of de aangegeven functietoets is ingedrukt, doch er wordt niet gesprongen, er wordt alleen een vlag gezet. Indien later KEY(nr)ON wordt gegeven, en de vlag voor de betreffende functietoets is gezet, dan wordt alsnog naar de subroutine gesprongen.

Syntax: KILL "[<dev>:]<fnam>"
Soort: Statement met MSX2 uitbreiding.
Omschrijving: Verwijdert het bestand met de naam <fnam> van de in <dev> aangegeven schijf. Voor MSX2 geldt, dat <dev> ook de RAM-disk kan zijn.
In <fnam> mogen ook vraagtekens en asterisken worden gebruikt, om meerdere bestanden tegelijkertijd te verwijderen. <dev> kan A, B, C, D, E, F of MEM zijn.

Syntax: LEFT$(<string>,<positie>)
Soort: Functie.
Omschrijving: Levert het linker deel van de in <string> gegeven tekens, tot en met de aangegeven <positie> op.

Syntax: LEN(<string>)
Soort: Functie.
Omschrijving: Levert een waarde op, die de lengte (in aantal tekens) van <string> aangeeft.

Syntax: LET<var>[$]=<uitdrukking>
Soort: Statement.
Omschrijving: Hiermee wordt het resultaat van <uitdrukking> toegekend aan de opgegeven variabele.
Indien een numerieke variabele wordt gebruikt, moet ook een numerieke uitdrukking worden gebruikt.

Syntax: LFILES ["[<dev>:][<fnam>]"]
Soort: Commando.
Omschrijving: Werkt als het commando FILES, alleen wordt nu de informatie op de aangesloten printer afgedrukt.

Syntax:
```
LINE [[STEP](<x1>,<y1>)]
     -[STEP](<x2>,<y2>)
     [,[<kl>][,[B[F]][,<logop>]]]
```
Soort: Statement met MSX2 uitbreiding.
Omschrijving: Tekent een lijn van coordinaat x1,y1 naar x2,y2 in de kleur <kl>.
Indien de parameter B wordt opgenomen, zal een rechthoek worden getekend,

waarvan de coordinaten x1,y1 en x2,y2 de diagonaal vormen.
Indien behalve de B ook parameter F wordt opgenomen, zal het getekende blok worden gevuld met de opgegeven kleur.

x1 en x2 - voor SCREEN 2, 3, 4, 5 en 8 een waarde van 0 - 255.
- voor SCREEN 6 en 7 een waarde van 0 - 511.

y1 en y2 - voor SCREEN 2, 3 en 4 een waarde van 0 - 191.
- voor SCREEN 5, 6, 7 en 8 een waarde van 0 - 211.

STEP - geeft aan dat de volgende x en y waarde een verplaatsing t.o.v. de laatste cursorpositie bevatten.

kl - voor SCREEN 2, 3, 4, 5 en 7 een geheel getal van 0 - 15.
voor SCREEN 6 een geheel getal van 0 - 3.
voor SCREEN 8 een geheel getal van 0 - 255.

logop - logische bewerking: AND, OR, XOR, PSET, PRESET, TAND, TOR, TXOR, TPSET en TPRESET.
De logische bewerking wordt uitgevoerd op de kleurnummers van de lijnkleur en de kleur van het scherm op de plaats waar de lijn wordt getrokken.
Indien <logop> wordt weggelaten wordt gewoon een lijn getrokken in de gegeven kleur

Syntax: LINE INPUT [<string>;]<string var.>
Soort: Statement.
Omschrijving: Leest alfanumerieke invoer vanaf het toetsenbord en plaatst dit in de opgegeven stringvariabele.
In de ingevoerde tekst mogen ook komma's voorkomen. Pas na het intikken van de RETURN-toets is de string beeindigd.
De maximum lengte van de ingevoerde string is 254 tekens.

Syntax: LINE INPUT #<filenummer>,<string var.>
Soort: Statement.
Omschrijving: Leest een record uit de file die onder <file nummer> was geopend voor INPUT.

Syntax: LIST [<beginregel>[-[<eindregel>]]]
Soort: Commando.
Omschrijving: Drukt alle BASIC-programmaregels van <beginregel> t/m <eindregel> af op het beeldscherm.
Indien <eindregel wordt weggelaten, worden alle regels van <beginregel> tot het einde van het programma afgedrukt.
Indien het koppelteken en <eindregel> worden weggelaten, wordt alleen <beginregel> afgedrukt.
Indien alle parameters worden weggelaten dan worden alle programmaregels afgedrukt.

Syntax: LLIST [<beginregel>[-[<eindregel>]]]
Soort: Commando.
Omschrijving: Doet precies hetzelfde als LIST, doch de output gaat naar de aangesloten printer.

Syntax: LOAD "<dev>:<file-naam>"[,R]
Soort: Statement met MSX2 uitbreiding.
Omschrijving: Laadt het opgegeven BASIC-programma in het geheugen.
dev = CAS, A, B, C, D, E, F, COM<nr> en voor MSX2 bovendien MEM.
nr = communicatiepoortnummer 0, 1 of 2 defaultwaarde = 0.
R = Indien deze optie is opgegeven, zal het programma automatisch worden gestart na het laden.

Syntax: LOC(<file-nummer>)
Soort: Functie.
Omschrijving: Levert de positie binnen de opgegeven disk-file op. Bij een sequentieel bestand het aantal gelezen of geschreven sectoren sinds het bestand werd geopend, bij een random bestand het aantal records.

Het file-nummer is aan het bestand toegekend tijdens het openen van het bestand.

Syntax: LOCATE [<hor.>[,<vert.>[,<cursor>]]]
Soort: Statement met MSX2 uitbreiding.
Omschrijving: Stuurt de cursor naar de opgegeven schermpositie.
<hor.> = horizontale positie (0 t/m 39) (voor MSX2 0 t/m 79)
<vert.> = verticale positie (0 t/m 23)
<cursor>= 0 - cursor niet zichtbaar, ten zij op input wordt gewacht.
1 - cursor zichtbaar.

Syntax: LOF(<file-nummer>)
Soort: Functie.
Omschrijving: Levert de lengte van het opgegeven bestand op in aantal bytes.
Het file-nummer is aan het bestand toegekend tijdens het openen van het bestand.

Syntax: LOG (<numerieke uitdrukking>)
Soort: Functie.
Omschrijving: Geeft de natuurlijke logaritme (grondtal e=2.718282) van de numerieke uitdrukking. De Briggse logaritme (grondtal 10) kan als volgt worden verkregen:
0.43429 * LOG(numerieke uitdrukking).

Syntax: LPOS(0)
Soort: Functie.
Omschrijving: Geeft de printpositie binnen het printerbuffer. Afhankelijk van de gebruikte printer hoeft dit geen indicatie te zijn van de werkelijk afgedrukte tekens.

Syntax: LPRINT [[USING <formaat>;]<uitdr.>...]
Soort: Statement.
Omschrijving: Werkt net als het overeenkomstige PRINT statement, doch drukt de output af op een aangesloten printer.

Syntax: LSET <X$>=<Y$>
Soort: Statement.
Omschrijving: Vult de variabele X$ van links naar rechts met de data uit de variabele of alfanumerieke uitdrukking Y$. Dit statement moet worden gebruikt voor het vullen van het buffer voor random disk bestanden.

Syntax: MAXFILES=<aantal>
Soort: Statement.
Omschrijving: Bepaalt het maximum aantal bestanden dat tegelijkertijd mag zijn geopend in een programma.
<aantal> = 0 t/m 15.

Syntax: MERGE "<dev>:<file-naam>"
Soort: Commando met MSX2 uitbreiding.
Omschrijving: Leest de programmaregels uit het opgegeven bestand en voegt deze regels toe aan het reeds in het geheugen staande programma. Indien dezelfde regel nummers voorkomen, zullen de in het geheugen staande regels worden overschreven met de nieuw ingelezen regels.
dev = CAS, A, B, C, D, E, F, COM<nr> en voor MSX2 bovendien MEM.
nr = communicatiepoortnummer 0, 1 of 2 defaultwaarde = 0.

Syntax: MID$(<string>,<vanaf>[,<aantal>])
Soort: Functie.
Omschrijving: Levert een string op die bestaat uit een deel van de opgegeven string. Dat deel start op positie <vanaf> en heeft een lengte <aantal>. Indien <aantal> wordt weggelaten, zal de rest van de string worden genomen. Indien <vanaf> voorbij het laatste teken van <string> wijst, zal een lege string ("") het resultaat zijn.

Syntax: MID$(<str1>,<start>[,<lengte>])=<str2>
Soort: Statement
Omschrijving: Vervangt het deel van string 1 (str1), dat wordt aangegeven met <start> en <lengte>, door tekens uit string 2 (str2). Indien geen lengte wordt opgegeven, zullen zoveel tekens worden vervangen als de lengte van string 1 dat toelaat.

Syntax: MKD$(<double precision-waarde>)
Soort: Functie
Omschrijving: Converteert de numerieke expressie van dubbele nauwkeurigheid naar een 8 bytes alfanumerieke waarde, die in het buffer voor een random disk-file kan worden geplaatst.

Syntax: MKI$(<integer-waarde>)
Soort: Functie
Omschrijving: Converteert de numerieke expressie van een geheel getal (integer) naar een 2 bytes alfanumerieke waarde, die in het buffer voor een random disk-file kan worden geplaatst.

Syntax: MKS$(<single precision-waarde>)
Soort: Functie
Omschrijving: Converteert de numerieke expressie van enkelvoudige nauwkeurigheid naar een 4 bytes alfanumerieke waarde, die in het buffer voor een random disk-file kan worden geplaatst.

Syntax: MOTOR [ON][OFF]
Soort: Statement
Omschrijving: Schakelt de motor van de cassette recorder aan of uit. Indien de parameters ON en OFF beide worden weggelaten, wordt de motor uitgeschakeld indien deze aan stond en aangeschakeld indien deze uit stond.

Syntax: NAME "<fnam-oud>" AS "<fnam-nieuw>"
Soort: Statement
Omschrijving: Geeft een nieuwe naam (fnam-nieuw) aan een bestaande disk-file (fnam-oud). De file-naam heeft het volgende formaat:

[<dev>:]<file-naam>[.<extentie>]

Indien <dev> wordt weggelaten, zal de laatst gebruikte schijf worden gebruikt. <dev> = A, B, C, D, E of F.

Syntax: NEW
Soort: Commando
Omschrijving: Wist het geheugen. Dient te worden gebruikt voordat een nieuw programma in het geheugen wordt geladen.

Syntax: OCT$(<decimale waarde>)
Soort: Functie
Omschrijving: Geeft de octale weergave van het decimale getal. Het decimale getal mag een waarde hebben van -32768 t/m 32767. Indien de waarde negatief is, zal de two's-complement notatie worden gebruikt.

Syntax: ON ERROR GOTO <regelnummer>
Soort: Statement
Omschrijving: Maakt dat er naar het aangegeven regelnummer wordt gesprongen zodra er een fout (syntax-fout of programmafout) wordt gedetecteerd. Op het aangegeven regelnummer dient de foutafhandelingsroutine te worden gestart. Indien regelnummer 0 wordt ingevuld, zal niet worden gesprongen, doch alleen de gebruikelijke foutboodschap worden gegeven. (Zie ook RESUME)

Syntax: ON <uitdr> GOSUB <regelnr>[,<regelnr>...]
Soort: Statement
Omschrijving: Start de subroutine die start op een aangegeven regelnummer. Er kunnen meerdere regelnummers achter het GOSUB-statement staan. Indien <uitdr> de waarde 1 heeft, zal het eerst gespecificeerde regelnummer worden gestart, een waar-

de 2 start het tweede regelnummer, etc. Indien <uitdr> een waarde heeft, waarvoor geen regelnummer aanwezig is, zal gewoon door het statement heengezakt worden.

Syntax: ON <uitdr> GOTO <regelnr>[,<regelnr>...]
Soort: Statement
Omschrijving: Werkt net zoals ON GOSUB (zie daar), doch er wordt naar een regelnummer gesprongen, in plaats van dat er een subroutine wordt gestart.

Syntax: ON INTERVAL=<waarde> GOSUB <regelnr>
Soort: Statement
Omschrijving: Nadat de interval-timing is geactiveerd met INTERVAL ON, zal met ON INTERVAL= de subroutine die op het aangegeven regelnummer staat worden gestart zodra een tijdspanne van <waarde>/50 is gepasseerd

Syntax: ON KEY GOSUB <regelnr>[,<regelnr>...]
Soort: Statement
Omschrijving: Nadat de functietoetsen zijn geactiveerd met het statement KEY(<nummer>) ON, zal met ON KEY GOSUB afhankelijk van de waarde van de ingedrukte functietoets een van de subroutines, waarvan de regelnummers zijn gegeven, worden gestart. (Werkt verder net zoals het ON GOSUB statement>)

Syntax: ON SPRITE GOSUB <regelnummer>
Soort: Statement
Omschrijving: Nadat het detecteren van "sprite collision" is geactiveerd met het statement SPRITE ON, zal op het botsen van twee sprites de aangegeven subroutine worden gestart.

Syntax: ON STOP GOSUB <regelnummer>
Soort: Statement
Omschrijving: Start de subroutine die op het aangegeven regelnummer begint, zodra de toetsen CONTROL en STOP tegelijkertijd worden ingedrukt.

Syntax: ON STRIG GOSUB <regelnr>[,<regelnr>...]
Soort: Statement
Omschrijving: Nadat de actieknoppen van de "hand-controls" (joy-stick, muis, etc.) zijn geactiveerd met STRIG(<nummer>) ON, kan met ON STRIG GOSUB een van de subroutines, die beginnen op de achter het statement opgenomen regelnummers, worden gestart. (Zie ook STRIG() ON.)

Syntax: OPEN "<fnam>" [FOR <mode>] AS [#]<nr> [LEN=<bytes>]
Soort: Statement met MSX2-uitbreiding
Omschrijving: Creeert een I/O-buffer voor een file.
De syntax voor <fnam> is als volgt:
[<dev>:]<file-naam>.<extentie>

<dev> =CRT voor het tekstscherm.
GRP voor het grafisch scherm.
LPT voor de printer.
CAS voor de cassetterecorder.
A t/ F voor schijven.
COM voor de RS232C-poort.
Bovendien MEM voor de RAM-disk in MSX2-computers.

<mode> =INPUT, OUTPUT of APPEND.
Indien geen mode wordt opgegeven, zal voor CRT, GRP en LPT mode OUTPUT gelden, voor CAS zal APPEND gelden, voor MEM en disk geldt RANDOM.

<nr> =Onder dit nummer kan de file met andere I/O-statements worden benaderd. Het hoogst toegestane nummer hangt af van MAXFILES, doch kan nooit hoger zijn dan 15 (default = 1).

<bytes>=De lengte van de records in een RANDOM-file, gemeten in bytes.

Het heeft alleen zin dit op te nemen indien een RANDOM-file wordt geopend. De maximale record-lengte is 256 bytes.

Syntax: OUT <I/O-poort>,<waarde>
Soort: Statement
Omschrijving: Schrijft een waarde naar de aangegeven I/O-poort. Dit statement maakt dat BASIC-programma's niet meer gegarandeerd werken op alle typen MSX-computers. Het is daarom beter dit statement niet te gebruiken.

Syntax: PAD(<nummer>)
Soort: Functie met MSX2-uitbreiding
Omschrijving: Levert de status van een tekenbord, lichtpen, muis of "track ball", die is aangesloten op een van de joystick-connectors, op. Welk van deze apparaten wordt gelezen hangt af van het gekozen nummer:

0 - 3 = tekenbord op joystick-poort 1.
0 = status van het tekenbord:
  0 = tekenbord niet aangeraakt.
  -1 = tekenbord aangeraakt.
1 = X-coordinaat van het tekenbord.
2 = Y-coordinaat van het tekenbord.
3 = schakelaar van het tekenbord:
  0 = schakelaar niet ingedrukt.
  -1 = schakelaar ingedrukt.
4 - 7 = tekenbord op joystick-poort 2.
4 = status van het tekenbord:
  0 = tekenbord niet aangeraakt.
  -1 = tekenbord aangeraakt.
5 = X-coordinaat van het tekenbord.
6 = Y-coordinaat van het tekenbord.
7 = schakelaar van het tekenbord:
  0 = schakelaar niet ingedrukt.
  -1 = schakelaar ingedrukt.

Uitbreiding voor MSX2:
8 - 11 = lichtpen.
8 = status van de lichtpen:
  0 = geen coordinaten beschikbaar.
  -1 = coordinaten beschikbaar.

9 = X-coordinaat van lichtpen.
10 = Y-coordinaat van lichtpen.
11 = schakelaar op lichtpen:
  0 = schakelaar niet ingedrukt.
  -1 = schakelaar ingedrukt.
12 - 15 = muis of track-ball op poort 1
12 = status (is altijd -1).
13 = X-coordinaat.
14 = Y-coordinaat.
15 = altijd de waarde 0.
16 - 19 = muis of track-ball op poort 2.
16 = status (is altijd -1).
17 = X-coordinaat.
18 = Y-coordinaat.
19 = altijd de waarde 0

Syntax: PAINT [STEP](x,y)[,[<inkt>][,<rand>]]

Soort: Statement met MSX2-uitbreiding.

Omschrijving: Vult een grafische figuur met de opgegeven "inkt"-kleur. De grafische figuur moet zijn getekend in de "rand"-kleur. De coordinaten x en y moeten binnen de grafische figuur liggen.
De optionele parameter STEP maakt de x en y coordinaten relatief ten opzichte van de laatst opgegeven coordinaten. Bij gebruik van STEP mogen x en y ook negatieve waarden zijn.

x = mag voor SCREEN 2, 3, 4, 5 en 8 een geheel getal van 0 tot 255 zijn.
mag voor SCREEN 6 en 7 een geheel getal van 0 tot 511 zijn.

y = mag voor SCREEN 2, 3 en 4 een geheel getal van 0 tot 191 zijn.
mag voor SCREEN 5, 6, 7 en 8 een geheel getal van 0 tot 211 zijn.

<inkt> en <rand>
voor SCREEN 2, 3, 4, 5 en 7 een geheel getal van 0 tot 15.
voor SCREEN 6 een geheel getal van 0 tot 3.
voor SCREEN 8 een geheel getal van 0 tot 255.

Syntax: PDL(<nummer>)
Soort: Functie.
Omschrijving: Leest de status van een op een joystick connector aangesloten games paddle. Deze status kan een waarde tussen 0 en 255 hebben. Met <nummer> wordt een bepaald deel van de games paddle geselecteerd.

<nummer> = 1, 3, 5, 7, 9 of 11:
games paddle aangesloten op joystick-connector 1.

<nummer> = 2, 4, 6, 8, 10 of 12:
games paddle aangesloten op joystick-connector 2.

Syntax: PEEK(<adres>)
Soort: Functie
Omschrijving: Leest de inhoud van het aangegeven geheugenadres. Dit kan elk adres van 0 tot 65535 zijn. De inhoud kan een waarde van 0 tot 255 hebben.

Syntax: PLAY <A$>[,[<B$>][,<C$>]]
Soort: Statement.
Omschrijving: Speelt muziek, zoals is aangegeven in de subcommando strings A$, B$ en C$, die respectievelijk aan de geluidskanalen A, B en C zijn toegewezen. De subcommando strings kunnen zijn samengesteld uit de volgende subcommando's:

T<X> Zet het tempo. <X> mag een waarde van 32 t/m 255 zijn. Het tempo is X/4 noten per minuut.

L<X> Zet de lengte van een noot. <X> mag een waarde van 1 t/m 64 zijn. De lengte van een noot is 1/X.

. Verandert de lengte van een noot. Iedere punt die een noot volgt, deelt de lengte daarvan door twee.

O<X> Selecteert een octaaf. <X> mag een waarde van 1 t/m 8 hebben.

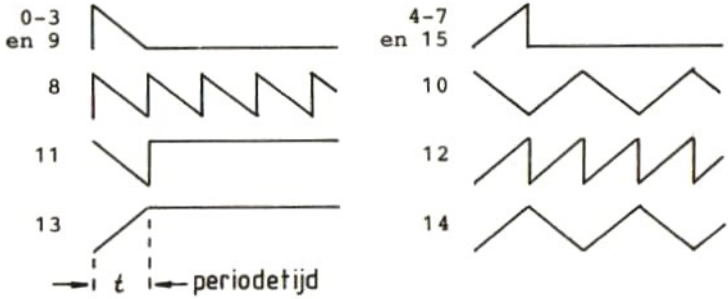
A/G Speelt de noten A, B, C, D, E, F of G. Iedere noot kan worden gevolgd door een + of # om een verhoogde, of door een - om een verlaagde noot aan te geven.

N<X> Speelt een noot, die is aangegeven door een nummer (X). X mag een waarde van 0 t/m 95 hebben.

R<X> Zet een rust. <X> mag een waarde hebben van 1 t/m 64. De duur van de rust is 1/X.

V<X> Zet het volume. <X> mag een waarde hebben van 0 t/m 16. Volume 16 schakelt de envelope-generator in.

S<X> Zet de vorm van de "sound envelope" <X> mag een waarde hebben van 0 t/m 15. Daarmee worden de volgende vormen gekozen:



M<X> Zet de periodetijd van de "sound envelope". <X> mag een waarde van 1 t/m 65535 hebben. De default waarde is 255.

"X<X$>;" Voert de subcommando's, die in variabele X$ staan, uit. Voorbeeld: X$="CDE":PLAY "XX$;"

Syntax: PLAY(<kanaal>)

Soort: Functie.

Omschrijving: Geeft aan of de geluidsopdracht voor het aangegeven kanaal al is uitgevoerd. <kanaal> kan 0, 1, 2 of 3 zijn, voor respectievelijk alle kanalen, kanaal A, kanaal B of kanaal C. Indien het resultaat een waarde 0 is, dan is de

geluidsopdracht uitgevoerd. De waarde -1 geeft aan dat de geluidsopdracht nog in uitvoering is.

Syntax: POINT(<hor>,<ver>)
Soort: Functie met MSX2-uitbreiding.
Omschrijving: Geeft, in grafische modes het kleurnummer van het aangegeven pixel.
<hor> = 0-255 voor SCREEN 2,3,4,5 en 8.
0-511 voor SCREEN 6 en 7.
<ver> = 0-191 voor SCREEN 2, 3 en 4.
0-211 voor SCREEN 5, 6, 7 en 8.

Syntax: POKE <adres>,<waarde>
Soort: Statement.
Omschrijving: Schrijft de <waarde> naar het <adres>.
<waarde> = 0 - 255.
<adres> = 0 - 65535.

Syntax: POS(0)
Soort: Functie met MSX2-uitbreiding.
Omschrijving: Geeft, in tekstmode 1 en 2, de horizontale positie van de cursor. Het resultaat kan liggen tussen 0 en 39, doch voor MSX2 kan het resultaat tussen 0 en 79 liggen.

Syntax: PRESET [STEP](<hor>,<ver>)[,<kl>][,<bew>]
Soort: Statement met MSX2-uitbreiding.
Omschrijving: Geeft in de grafische modes het pixel op de coordinaten <hor>,<ver> de kleur <kl>. Het al of niet toekennen van de kleur kan worden beinvloed door middel van de logische bewerking <bew>.
Indien de optionele parameter STEP wordt gebruikt, zullen de coordinaten worden beschouwd als relatieve coordinaten ten opzichte van het laatst gebruikte pixel.
<hor>= 0-255 voor SCREEN 2,3,4,5 en 8.
0-511 voor SCREEN 6 en 7.
<ver>= 0-191 voor SCREEN 2, 3 en 4.
0-211 voor SCREEN 5, 6, 7 en 8.
<kl> = 0-15 voor SCREEN 2,3,4,5 en 7.
0-3 voor SCREEN 6.
0-255 voor SCREEN 8.

Indien geen kleurnummer wordt opgegeven, wordt het laatstgebruikte kleurnummer gebruikt.

<bew>= AND, OR, PSET, PRESET, XOR, TAND, TOR, TPSET, TPRESET, TXOR.
Indien de logische bewerking wordt weggelaten, zal PSET worden aangenomen. De logische bewerkingen worden uitgevoerd op de binaire waarden van <kl> en de kleur van het aangegeven pixel. De letter T voor de logische bewerking wil zeggen dat transparante kleuren geen effect hebben.

Syntax: PRINT [[USING <formaat>;]<expr>;...]
PRINT #<x>,[USING <formaat>;]<var>;...

Soort: Statement

Omschrijving: Het eerste formaat drukt gegevens uit <expr> af op het beeldscherm, indien gewenst met gebruikmaking van het opgegeven <formaat>.
Het tweede formaat drukt gegevens uit <var> af naar sequentiele file nummer <x>. De file moet zijn geopend voor output. Indien gewenst kan gebruik worden gemaakt van het opgegeven <formaat>.
De expressies kunnen van elkaar worden gescheiden en het PRINT-statement kan worden afgesloten door een van de volgende tekens:

- spatie - Het volgende gegeven wordt op de volgende regel gedrukt.
- ; - Het volgende gegeven wordt direct na dit gegeven afgedrukt
- , - Het volgende gegeven wordt aan het begin van de volgende kolom afgedrukt. Iedere kolom is 14 tekens breed.

Op het scherm afgedrukte getallen worden altijd gevolgd door een spatie. Positieve getallen worden ook voorafgegaan door een spatie. Negatieve getallen worden voorafgegaan door een min-teken.

Wanneer naar een file wordt geprint, zal iedere PRINT-opdracht worden afgesloten met een CR/LF-code. Tussen de numerieke expressies worden automatisch komma's gezet. Tussen alfanumerieke expressies moet men zelf een komma zetten als scheidingsteken.

USING <formaat>

Alfanumerieke formaten:
USING "!"
Alleen het eerste teken van de gegeven expressie wordt afgedrukt.
USING "\ \"
Het aantal tekens van de expressie die worden afgedrukt bedraagt 2 plus het aantal spaties tussen de \-tekens.
USING "@"
Het @-teken wordt vervangen door de gegeven expressie.

Numerieke formaten:
USING "#"
Geeft het aantal cijfers van het getal aan. Het getal zal naar rechts uitgelijnd worden. Links wordt dan aangevuld met spaties. Zonodig wordt het getal afgerond.
USING "."
De punt geeft de plaats van de decimale komma aan.
USING ","
Een komma links van de decimale punt geeft aan, dat er om de drie cijfers een komma zal worden afgedrukt.
USING "+"
Een plus-teken aan het begin van het formaat geeft aan dat een plus- of minteken voor het getal zal worden afgedrukt. Staat het plus-teken achter het formaat, dan wordt plus- of min-teken achter het getal afgedrukt.
USING "-"
Een min-teken achteraan het formaat

heeft tot gevolg dat achter een positief getal een spatie en achter een negatief getal een min-teken wordt afgedrukt.
USING "**"
Voorlopende nullen worden vervangen door asterisken in plaats van spaties.
USING "$$"
Het getal wordt voorafgegaan door een dollar-teken.
USING "^^^^"
Het getal zal in exponentiele vorm worden afgedrukt.

Syntax: PSET [STEP](<hor>,<ver>)[,<kl>][,<bew>]
Soort: Statement met MSX2-uitbreiding.
Omschrijving: Geeft in de grafische modes het pixel op de coordinaten <hor>,<ver> de kleur <kl>. Het al of niet toekennen van de kleur kan worden beinvloed door middel van de logische bewerking <bew>.
Indien de optionele parameter STEP wordt gebruikt, zullen de coordinaten worden beschouwd als relatieve coordinaten ten opzichte van het laatst gebruikte pixel.
<hor>= 0-255 voor SCREEN 2,3,4,5 en 8.
0-511 voor SCREEN 6 en 7.
<ver>= 0-191 voor SCREEN 2, 3 en 4.
0-211 voor SCREEN 5, 6, 7 en 8.
<kl> = 0-15 voor SCREEN 2,3,4,5 en 7.
0-3 voor SCREEN 6.
0-255 voor SCREEN 8.
Indien geen kleurnummer wordt opgegeven, wordt het laatstgebruikte kleurnummer gebruikt.
<bew>= AND, OR, PSET, PRESET, XOR, TAND, TOR, TPSET, TPRESET, TXOR.
Indien de logische bewerking wordt weggelaten, zal PSET worden aangenomen. De logische bewerkingen worden uitgevoerd op de binaire waarden van <kl> en de kleur van het aangegeven pixel. De letter T voor de logische bewerking wil zeggen dat transparante kleuren geen effect hebben.

Syntax: PUT [#]<filenr>[,<recordnr>]

Soort: Statement.

Omschrijving: Schrijft een record onder het aangegeven nummer van het buffer naar de aangegeven random file. Het recordnummer mag varieren van 0 tot 32767. Indien geen recordnummer wordt opgegeven zal het volgende recordnummer worden gebruikt.

Syntax: PUT SPRITE <priority>[,[STEP}(<X>,<Y>)] [,<kleur>][,<spritenr>]

Soort: Statement met MSX2 uitbreiding.

Omschrijving: Plaatst de aangegeven sprite op de coordinaten X en Y, in de aangegeven kleur. Indien geen kleur wordt opgegeven blijft de sprite de kleur(en) behouden die er met COLOR SPRITE aan waren toegekend. In SCREEN 2 t/m 4 kunnen slechts vier sprites op dezelfde regel worden geplaatst, in SCREEN 5 t/m 8 kunnen acht sprites op dezelfde regel worden geplaatst.

<X> = -32 tot en met 255
<Y> = -32 tot en met 191 of 208 of 209.
- 208 = alle sprites met lagere prioriteit verdwijnen.
- 209 = alleen deze sprite wordt onzichtbaar.

<priority> = 0 tot en met 31.
<kleur> = 0 tot en met 15.
<spritenr> = Het aan deze sprite met SPRITE$(X) toegekende spritenummer. Indien <spritenr> wordt weggelaten, zal het prioriteitsnummer worden gebruikt.

Syntax: READ <var>[,<var>...]

Soort: Statement.

Omschrijving: Leest gegevens uit DATA-regels in de variabele(n). Indien de variabele numeriek is, zal het gelezen item ook numeriek dienen te zijn. Is de variabele alfanumeriek, dan moet het gelezen item ook alfanumeriek zijn. Indien er geen

gegevens meer zijn om te lezen, zal de boodschap "out of data" worden gegeven.

Syntax:	REM <commentaar>
Soort:	Statement.
Omschrijving:	REM-statements worden alleen afgedrukt bij het maken van een listing. Ze worden niet uitgevoerd. De functie van een REM-statement is, de listing van het programma duidelijker te maken voor de programmeur. In plaats van de letters REM mag ook het teken ' (apostrophe) worden gebruikt.

Syntax:	RENUM [[<nieuw>][,<oud>][<ophoging>]]
Soort:	Commando.
Omschrijving:	Hernummert alle regelnummers vanaf het regelnummer <oud>. De nieuwe regelnummers gaan vanaf regelnummer <nieuw>, waarbij iedere volgende regel een nummer heeft dat <ophoging> hoger is dan het vorige. Indien <oud> wordt weggelaten, zal het programma vanaf de eerste regel worden hernummerd. Indien <ophoging> wordt weggelaten, zal de ophogingsfactor 10 zijn. Indien geen enkele parameter wordt opgegeven, zal het hele programma worden hernummerd, waarbij de nieuwe regelnummers starten op 10 met een ophogingsfactor van 10.

Syntax:	RESTORE [<regelnr>]
Soort:	Statement.
Omschrijving:	Zet de "READ-pointer" op het eerste element van DATA-regel <regelnr>. Indien geen regelnummer wordt opgegeven zal de "READ-pointer" op het eerste element van de eerste DATA-regel binnen het programma worden gezet.

Syntax:	RIGHT$(<string>,<aantal>)
Soort:	Functie.
Omschrijving:	Geeft het <aantal> meest rechtse tekens van de opgegeven <string>. Indien <aantal> groter is dan het aantal tekens in de string, wordt de gehele string genomen. Indien <aantal> nul is, zal deze functie een "empty-string" opleveren.
Syntax:	RND(<waarde>)
Soort:	Functie.
Omschrijving:	Leest een getal uit een denkbeeldige tabel, waarin getallen tussen 0 en 1 in een willekeurige volgorde staan. Door <waarde> kan de plaats binnen de tabel, die wordt gelezen, worden beinvloed. <waarde> = 0: Het laatste nummer uit de tabel. <waarde> = positief: Het volgende nummer uit de tabel. <waarde> = negatief getal: Het te lezen nummer hangt af van de waarde van het negatieve getal.
Syntax:	RSET <doel-string>=<bron-string>
Soort:	Statement.
Omschrijving:	Zet de inhoud van de bron-string in de doel-string, beginnende aan de rechter kant. Dit statement dient te worden gebruikt om data in het random diskette buffer te zetten.
Syntax:	RUN [<regelnr>] RUN [<per>:]<prognaam>[,R]
Soort:	Commando met MSX2 uitbreiding.
Omschrijving:	Het eerste formaat start een in het geheugen staand programma, eventueel op de aangegeven regel. Het tweede formaat laadt het aangegeven programma van het aangegeven device. De optie R voorkomt dat nog openstaande bestanden worden gesloten. <dev>=CAS, A t/m F, COM of MEM. MEM is alleen mogelijk in MSX2-computers

Syntax: SAVE "<dev>:<file-naam>"[,A]
Soort: Statement met MSX2 uitbreiding.
Omschrijving: Schrijft het opgegeven BASIC-programma vanuit het geheugen naar het aangegeven device.

dev = CAS, A, B, C, D, E, F, COM<nr> en voor MSX2 bovendien MEM.
Indien <dev> = CAS, dan wordt het programma als ASCII-file naar cassette geschreven.
Voor de andere devices geldt dat het programma in "compressed binary" wordt weggeschreven, tenzij de optie A aan het statement wordt toegevoegd.

nr = communicatiepoortnummer 0, 1 of 2 defaultwaarde = 0.

A = Indien deze optie is opgegeven, zal het programma als ASCII-file worden weggeschreven.

Syntax: SCREEN [[<mode>],[<sprite>],[<klik>],[<baud>],[< MSX>],[<disp>]]
Soort: Statement met MSX2 uitbreiding.
Omschrijving: Stelt beeldschermmode, spritegrootte, toetsenbordklik, cassetteschrijfsnelheid printersoort en weergavemode in.

<mode> = 0:tekstmode 24 regels van 40 of 80 tekens (stel in met WIDTH).
1:tekstmode 24 regels van 32 tekens.
2:grafische mode 256*192 pixels in 16 kleuren.
3:grafische mode 64*48 blokken in 16 kleuren.
+ voor MSX2:
4:grafische mode 256*192 pixels in 16 kleuren.
5:grafische mode 256*212 pixels in 16 kleuren.
6:grafische mode 512*212 pixels in 4 kleuren.
7:grafische mode 512*212 pixels in 16 kleuren.
8:grafische mode 256*212 pixels

in 256 kleuren.
<sprite>=0:sprites van 8*8 pixels
1:vergrootte 8*8 sprites van 16*16 pixels.
2:sprites van 16*16 pixels
3:vergrootte 16*16 sprites van 32*32 pixels.
<klik> =0:geen toetsklik hoorbaar.
1:toetsklik hoorbaar.
<baud> =1:cassette schrijfsnelheid is 1200 baud.
2:cassette schrijfsnelheid is 2400 baud.
<MSX> =0:MSX-printer.
1:Geen MSX-printer.
<disp> =0:normaal.
1:interlaced.
2:normaal, alternerend even/oneven.
3:interlaced, alternerend even/oneven.
Modes 2 en 3 zijn alleen mogelijk wanneer een oneven schermpaginanummer wordt gebruikt (zie SET PAGE).

Het SCREEN-statement wist alle sprites van het scherm.

Syntax: SET ADJUST(<hor>,<ver>)
Soort: MSX2-statement.
Omschrijving: Verplaatst het gehele beeldscherm in horizontale en verticale richting. De positie wordt vastgelegd in de klokchip.
<hor> = een waarde van -7 tot 8.
<ver> = een waarde van -7 tot 8.
De waarden slaan op het aantal pixels dat het beeld moet worden verplaatst.

Syntax: SET BEEP <toon>,<vol>
Soort: MSX2-statement.
Omschrijving: Wijzigt de toon(soort) en het volume van de pieptoon en slaat dit op in de klokchip.
<toon>=1:korte hoge beep.

2:gong-geluid.
3:twee-tonige beep.
4:drie-tonige beep.

<vol> =1 t/m 4 voor instelling van het volume van de beep.

Syntax: SET DATE "<datum>"[,A]
Soort: MSX2-statement.
Omschrijving: Stelt de datum in en bewaart deze in de klok-chip. Met de optionele parameter A kan een alarm-datum in de klok-chip worden ingesteld. Er is echter geen MSX2 statement om hiervan gebruik te maken.

<datum> = MM/DD/JJ of
DD/MM/JJ of
JJ/MM/DD, afhankelijk van het type MSX2-computer.

Alle cijfers moeten worden ingetikt, ook als het voorlopende nullen zijn.

Syntax: SET PAGE <page>,<act>
Soort: MSX2-statement.
Omschrijving: Stelt de pagina uit het video-geheugen die wordt weergegeven in en stelt de pagina uit het video-geheugen waar naartoe wordt geschreven of waaruit wordt gelezen in. Beide paginanummers worden in de klok-chip opgeslagen.

<page>= 0 t/m 3 (is paginanummer dat wordt weergegeven.)
<act> = 0 t/m 3 (is paginanummer waar I/O naartoe gaat.)

Syntax: SET PASSWORD "<wachtwoord>"
Soort: MSX2-statement.
Omschrijving: Stelt een wachtwoord in en slaat deze op in de klok-chip. Na aanschakelen van de computer dient dit wachtwoord in te worden getikt, voordat het systeem verder wordt opgestart.

Indien SET PASSWORD wordt gebruikt, kunnen SET PROMPT en SET TITLE niet meer worden gebruikt.

<wachtwoord>=maximaal 255 tekens.

Syntax: SET PROMPT "<woord>"
Soort: MSX2-statement.
Omschrijving: Wijzigt de "Ok"-prompt in de nieuwe met <woord> opgegeven prompt.
Indien SET PROMPT wordt gebruikt, kunnen SET PASSWORD en SET TITLE niet meer worden gebruikt.
<woord> = maximaal 6 tekens.

Syntax: SET SCREEN
Soort: MSX2-statement.
Omschrijving: Slaat de momentele scherm-instellingen op in de klok-chip. De volgende instellingen worden opgeslagen:
Schermmode,
Regellengte,
voorgrondkleur,
achtergrondkleur,
randkleur,
weergave van de functietoetsen,
toetsklik,
printersoort,
cassetteschrijfsnelheid,
weergavescherm.

Syntax: SET TIME "<tijd>",A
Soort: MSX2-statement.
Omschrijving: Stelt de tijd in en slaat deze op in de klok-chip. Met de optionele parameter A kan een alarmtijd in de klok-chip worden ingesteld, er is echter geen MSX2-statement om hiervan gebruik te maken.
<tijd> = UU:MM:SS
Alle cijfers moeten worden ingetikt, ook als het voorlopende nullen zijn.

Syntax: SET TITLE "<titel>"[,<kleur>]
Soort: MSX2-statement.
Omschrijving: Stelt een nieuwe titel in, die tijdens het opstarten van het systeem op het beeldscherm wordt afgedrukt en slaat deze titel op in de klok-chip.
<titel> = maximaal 6 tekens.
<kleur> = kleurnummer 1 t/m 4.
Indien SET TITLE wordt gebruikt, kunnen

SET PASSWORD en SET PROMPT niet meer worden gebruikt.

Syntax: SET VIDEO [<mode>],[<hel>],[<kl>],[<sync>],[<audio>],[<ext>],[<A/V>]

Soort: MSX2-statement.

Omschrijving: Stelt de "super-impose"-mode in, waardoor de MSX2-computer kan dienen als regelorgaan voor computer-graphics en video-beelden.

<mode> = 0 t/m 3
0 - beeld komt uit de computer.
1 - beeld komt uit de computer.
2 - computerbeeld over video-beeld.
3 - video-beeld.

<hel> = 0 of 1
0 - halve intensiteit.
1 - volle intensiteit.

<kl> = 0 of 1
0 - colour bus output.
1 - colour bus input.

<sync> = 0 of 1
0 - interne sync.
1 - externe sync.

<audio>= 0 t/m 3
0 - geluid uit de computer.
1 - rechter kanaal van externe audio gemixed met computer geluid.
2 - linker kanaal van externe audio gemixed met computer geluid.
3 - beide externe audio kanalen gemixed met computer geluid.

<ext> = 0 of 1
0 - externe video input via RGB-euroconnector.
1 - externe video input via TV-connector.

<A/V> = 0 of 1
0 - geen A/V output via RGB-connector.
1 - A/V output via RGB-connector.

Default-waarde voor alle parameters is 0 zodat de computer normaal wordt gebruikt

Syntax: SGN(<waarde>)
Soort: Functie.
Omschrijving: Bepaalt het teken van de opgegeven <waarde>. Het resultaat kan zijn:
1 waarde is groter dan 0
0 waarde is gelijk aan 0
-1 waarde is kleiner dan 0.

Syntax: SIN(<radialen>)
Soort: Functie.
Omschrijving: Berekent de sinus van het opgegeven aantal radialen.

Syntax: SOUND <register>,<inhoud>
Soort: Statement.
Omschrijving: Kent de opgegeven inhoud toe aan het opgegeven register van de sound-processor.
<register> = 0 t/m 13.
<inhoud> = 0 t/m 255.

Syntax: SPACE$(<aantal>)
Soort: Functie.
Omschrijving: Levert het opgegeven aantal spaties op.
<aantal> = 0 t/m 255.

Syntax: SPC(<aantal>)
Soort: Functie.
Omschrijving: Drukt het opgegeven aantal spaties af.
<aantal> = 0 t/m 255.

Syntax: SPRITE [ON][OFF][STOP]
Soort: Statement.
Omschrijving: Na SPRITE ON wordt na iedere instructie gecontroleerd of twee sprites met elkaar in botsing zijn gekomen en zo dit het geval is, zal de subroutine die met ON SPRITE GOSUB is opgegeven worden aangeroepen. Na SPRITE OFF wordt niet meer gecontroleerd op botsingen van sprites. Na SPRITE STOP wordt wel gecontroleerd op een botsing, doch de subroutine zal niet worden uitgevoerd. Pas nadat SPRITE ON is gegeven zal de subroutine alsnog worden uitgevoerd.

Syntax: SPRITE$(<nummer>)
Soort: Systeemvariabele.
Omschrijving: Hieraan kan de inhoud van een sprite worden toegekend. Indien er sprake is van kleine sprites, moeten er 8 codes die varieren in waarde van 0 tot 255 aan SPRITE$ worden toegekend. Indien er sprake is van grote sprites, moeten er 24 codes worden toegekend.
<nummer> = 0 t/m 255 en is het nummer waaronder de sprite later kan worden opgeroepen.

Syntax: SQR(<waarde>)
Soort: Functie.
Omschrijving: Berekent de vierkantswortel uit de opgegeven waarde.

Syntax: STICK(<nummer>)
Soort: Functie.
Omschrijving: Geeft de status van de cursor-toetsen of joysticks (zoals aangegeven met nummer).
<nummer> = 0 t/m 2
0 - cursor-toetsen.
1 - joystick 1.
2 - joystick 2.
De status kan een van de volgende waarden zijn:
0 - niets ingedrukt.
1 - boven.
2 - rechtsboven.
3 - rechts.
4 - rechtsonder.
5 - onder.
6 - linksonder.
7 - links.
8 - linksboven.

Syntax: STOP
Soort: Statement.
Omschrijving: Onderbreekt de uitvoering van het programma met de boodschap "Break in <regelnummer>". Het programma kan worden voortgezet met het commando CONT.

Syntax: STOP [ON][OFF][STOP]
Soort: Statement.
Omschrijving: Na STOP ON wordt na ieder statement gecontroleerd of de CONTROL en STOP toetsen tegelijkertijd zijn ingedrukt. Is dit het geval, dan wordt de met ON STOP GOSUB aangegeven subroutine gestart. Na STOP OFF wordt er niet meer op het indrukken van de toetsen gecontroleerd. Na STOP STOP wordt nog wel gecontroleerd, doch wordt de subroutine niet gestart. Na een STOP ON statement zal de subroutine dan alsnog worden gestart.

Syntax: STRIG(<nummer>)
Soort: Functie.
Omschrijving: Geeft de status van de spatiebalk of van de actieknoppen, zoals gekozen met <nummer>.
<nummer> = 0 t/m 4
0 - spatiebalk.
1 - 1e actieknop van "hand control 1"
2 - 1e actieknop van "hand control 2"
3 - 2e actieknop van "hand control 1"
4 - 2e actieknop van "hand control 2"
"hand control" kan een joystick, muis en dergelijke zijn.
De resulterende status kan als volgt zijn:
0 - actieknop of spatiebalk niet ingedrukt.
-1 - actieknop of spatiebalk ingedrukt

Syntax: STRIG(<nummer>)[ON][OFF][STOP]
Soort: Statement.
Omschrijving: Na STRIG(<nummer>)ON wordt na ieder statement gecontroleerd of de met <nummer> aangegeven actieknop is ingedrukt. Is dit het geval, dan wordt de met het statement ON STRIG GOSUB aangegeven subroutine gestart. Na STRIG (<nummer>)OFF wordt er niet meer op het indrukken van de betreffende toets gecontroleerd. Na STRIG(<nummer>)STOP

wordt nog wel gecontroleerd, doch wordt de subroutine niet meer gestart. Zodra een STRIG(<nummer>ON statement wordt uitgevoerd, zal de subroutine alsnog worden uitgevoerd.
<nummer> = 0 t/m 4
0 - spatiebalk.
1 - 1e actieknop van "hand control 1"
2 - 1e actieknop van "hand control 2"
3 - 2e actieknop van "hand control 1"
4 - 2e actieknop van "hand control 2"

Syntax: STR$(<getal>)
Soort: Functie.
Omschrijving: Geeft de numerieke waarde van <getal> weer als een alfanumerieke string.

Syntax: STRING$(<len>,[<code>][<var>])
Soort: Functie.
Omschrijving: Genereert een alfanumerieke string met de in <len> aangegeven lengte en met als inhoud tekens van de aangegeven <code> of van het eerste teken uit de aangegeven variabele.
<len> = 0 t/m 255.
<code> = 32 t/m 255.
<var> = naam van een string-variabele.

Syntax: SWAP <var1>,<var2>
Soort: Statement.
Omschrijving: Verwisselt de inhoud van <var1> en <var2>. Beide variabelen moeten van hetzelfde type zijn (numeriek of string)

Syntax: TAB(<pos>)
Soort: Functie.
Omschrijving: Verplaatst de cursor, op dezelfde regel, naar positie <pos>.
<pos> = 0 t/m 255.

Syntax: TAN(<radialen>)
Soort: Functie.
Omschrijving: Berekent de tangens van de in <radialen> opgegeven hoek.

Syntax: TIME
Soort: Systeemvariabele.
Omschrijving: Bevat de stand van de interne teller, die 50 keer per seconde met 1 wordt verhoogd.

Syntax: TROFF
Soort: Commando.
Omschrijving: Schakelt de trace-functie uit. (zie TRON).

Syntax: TRON
Soort: Commando.
Omschrijving: Schakelt de trace-functie in. Hierna zullen de regelnummers van alle regels die worden uitgevoerd worden afgedrukt. Dit maakt het mogelijk het verloop van het programma te volgen en te zien waar het eventueel fout gaat.

Syntax: USR[<nummer>](<param>)
Soort: Functie.
Omschrijving: Roept de aangegeven machinetaalroutine aan en geeft de parameter <param> vanuit BASIC door aan de machinetaalroutine.
Indien <nummer> wordt weggelaten, zal nummer 0 worden aangeroepen.
<nummer> = 0 t/m 9.
<param> kan een numerieke of alfanumerieke waarde zijn:

% - Integer waarde.
  Adres &HF663 bevat de waarde 2.
  Adressen &HF7F8 en &HF7F9 bevatten de integer-waarde.

$ - Alfanumerieke waarde.
  Adres &HF663 bevat de waarde 3.
  Adressen &HF7F8 en &HF7F9 bevatten het adres van een drie bytes descrip tor blok.
  Het eerste byte van dit blok geeft de lengte van de string, de volgende twee bytes geven het start adres van de string in het RAM-geheugen.

! - Enkele nauwkeurigheid.
  Adres &HF663 bevat de waarde 4.

    Adressen &HF7F6 t/m &HF7F9 bevatten
    de waarde met enkele nauwkeurigheid.
\# - Dubbele nauwkeurigheid.
    Adres &HF663 bevat de waarde 8.
    Adressen &HF7F6 t/m &HF7FD bevatten
    de waarde met dubbele nauwkeurigheid

Waarden die door de machinetaalroutine aan het BASIC-programma worden teruggegeven moeten in een variabele van het juiste type worden opgevangen.

Syntax: VAL(<string>)
Soort: Functie.
Omschrijving: Geeft de numerieke waarde van de alfanumerieke <string>.

Syntax: VARPTR([<var>][#<filenr>])
Soort: Functie.
Omschrijving: Geeft het adres van het eerste byte van de in <var> aangegeven variabele of het adres van het eerste byte van het file control block van de in <filenr> aangegeven file.
Het resultaat van de functie zal een waarde tussen -32768 en +32767 zijn. Indien de waarde negatief is moet er 65535 worden bijgeteld om het werkelijke adres te verkrijgen.

Syntax: VDP(<register>)
Soort: Systeemvariabele met MSX2-uitbreiding.
Omschrijving: Bevat de inhoud van het aangegeven besturingsregister van de Video Display Processor.
<register> = 0 t/m 8 (voor MSX1 en MSX2)
= -9 t/m -1, 9 t/m 24 en
33 t/m 47 voor MSX2.
Zie verder hoofdstuk 15, Video Display Processor.

Syntax: VPEEK(<adres>)
Soort: Functie met MSX2-uitbreiding.
Omschrijving: Leest de inhoud van het opgegeven videogeheugenadres.
<adres> = 0 t/m 16383 (SCREEN 0 t/m 4).

0 t/m 65535 (SCREEN 5 t/m 8).
Voor SCREEN 5 en 6 wordt het absolute video-geheugenadres als volgt berekend:
<actieve paginanummer>*&H8000+<adres>.
Voor SCREEN 7 en 8 wordt het absolute video-geheugenadres als volgt berekend:
<actieve paginanummer>*&H10000+<adres>.

Syntax: VPOKE <adres>,<inhoud>
Soort: Statement met MSX2-uitbreiding.
Omschrijving: Schrijft de <inhoud> naar het opgegeven video-geheugenadres.
<adres> = 0 t/m 16383 (SCREEN 0 t/m 4).
0 t/m 65535 (SCREEN 5 t/m 8).
Voor SCREEN 5 en 6 wordt het absolute video-geheugenadres als volgt berekend:
<actieve paginanummer>*&H8000+<adres>.
Voor SCREEN 7 en 8 wordt het absolute video-geheugenadres als volgt berekend:
<actieve paginanummer>*&H10000+<adres>.
<inhoud> = 0 t/m 255.

Syntax: WAIT <poort>,<xor>[,<and>]
Soort: Statement.
Omschrijving: Leest de momentele waarde van de aangegeven input-poort en vergelijkt deze waarde met de uitdrukking(en) <xor> en <and>. Deze vergelijkingen worden als volgt uitgevoerd:
<poort> XOR <xor>
<poort> AND <and>
Indien het resultaat van de vergelijking 0 is, wordt de vergelijking nogmaals uitgevoerd. Pas wanneer het resultaat ongelijk aan 0 is wordt het programma voortgezet.
Met dit statement wordt direct een poortnummer geadresseerd. Poortnummers zijn niet allemaal gestandaardiseerd, zodat een programma waarin dit statement wordt gebruikt, niet gegarandeerd goed werkt op alle MSX-computers.

Syntax: WIDTH <tekens>

Soort: Statement met MSX2-uitbreiding.

Omschrijving: Stelt de regellengte in op het aangegeven aantal tekens.

<tekens>= 1 t/m 32 (SCREEN 1, MSX1 en 2)
1 t/m 40 (SCREEN 0, MSX1 en 2)
80 (SCREEN 0, MSX2).

Het opgegeven aantal tekens zal gecentreerd worden afgedrukt op het scherm.

9 Systeemboodschappen

In dit hoofdstukje zullen de systeemboodschappen op drie manieren worden weergegeven.

1. Op alfabetische volgorde, gevolgd door het bijbehorend boodschapnummer.
2. Op volgorde van boodschapnummer, gevolgd door de boodschap zelf.
3. Op alfabetische volgorde, gevolgd door een nadere toelichting.

Mocht u tijdens het uitvoeren van een programma een systeemboodschap krijgen, dan kunt u uit de eerste lijst het bijbehorende boodschapnummer halen. Dit nummer kunt u gebruiken om in een foutafhandelingsroutine te gebruiken.

Tijdens het ontwikkelen van een programma, en speciaal tijdens het schrijven van de foutafhandelingsroutine daarin, kunt u zowel lijst 1 als lijst 2 gebruiken. Mocht u nadere informatie wensen over een systeemboodschap, dan zal lijst 3 u van dienst kunnen zijn.

Lijst 1

Systeemboodschap	nummer	Systeemboodschap	nummer
Bad drive name	62	File not found	53
Bad FAT	60	File not open	59
Bad file mode	61	File still open	64
Bad file name	56	Illegal direct	12
Bad file number	52	Illegal function call	5
Bad sector number	63	Input past end	55
Can't continue	17	Internal error	51
Device I/O error	19	Line buffer overflow	25
Direct statement	57	Missing operand	24
Disk error	(geen nr)	NEXT without FOR	1
Disk full	66	No RESUME	21
Disk I/O error	69	Out of DATA	4
Disk offline	70	Out of memory	7
Disk write protected	68	Out of string space	14
Division by zero	11	Overflow	6
Field overflow	50	Redimensioned array	10
File already exists	65	Rename across disk	71
File already open	54	RESUME without error	22

Systeemboodschap	nummer	Systeemboodschap	nummer
RETURN without GOSUB	3	Too many files	67
Sequential I/O only	58	Type mismatchs	13
String formula too complex	16	Undefined line nr.	8
String too long	15	Undefined user func.	18
Subscript out of range	9	Unprintable errors	23 26-49 72-255
Syntax error	2	Verify error	20

Lijst 2

Nr	Systeemboodschap	Nr	Systeemboodschap
1	NEXT without FOR	25	Line buffer overflow
2	Syntax error	26-49	Unprintable errors
3	RETURN without GOSUB	50	Field overflow
4	Out of DATA	51	Internal error
5	Illegal function call	52	Bad file number
6	Overflow	53	File not found
7	Out of memory	54	File already open
8	Undefined line number	55	Input past end
9	Subscript out of range	56	Bad file name
10	Redimensioned array	57	Direct statement
11	Division by zero	58	Sequential I/O only
12	Illegal direct	59	File not open
13	Type mismatch	60	Bad FAT
14	Out of string space	61	Bad file mode
15	String too long	62	Bad srive name
16	String formula too complex	63	Bad sector number
17	Can't continue	64	File still open
18	Undefined user func.	65	File already exists
19	Device I/O error	66	Disk full
20	Verify error	67	Too many files
21	No RESUME	68	Disk write protected
22	RESUME without error	69	Disk I/O error
23	Unprintable error	70	Disk off line
24	Missing operand	71	Rename across disk

Aan de nummers 72 tot en met 255 zijn nog geen boodschappen toegekend. De programmeur mag aan deze nummers zelf een boodschap koppelen.

Lijst 3

Bad drive name
De in het statement opgegeven diskette drive is niet in gebruik.

Bad FAT
De disketten waarop I/O wordt gewenst, is nog niet geformateerd.

Bad file mode
De statements PUT, GET of LOF zijn gebruikt op een sequentieel bestand.
Het statement LOAD is gebruikt op een "random access" bestand.
Het bestand wordt met een niet toegestane mode geopend.

Bad file name
De bestandsaanduiding bevat syntax fouten.

Bad file number
De aangegeven file is niet onder het aangegeven file nummer geopend, of het file nummer is hoger dan volgens "MAXFILES" is toegestaan.

Bad sector number
Het in PUT of GET gebruikte bloknummer is kleiner dan 1 of groter dan 32767.

Can't continue
Er is geen programma in het geheugen, of het programma werd onderbroken door een foutboodschap, of het programma werd, nadat het was onderbroken, gewijzigd.

Device I/O error
Tijdens het lezen of schrijven van gegevens of programma's is een lees- of schrijffout ontdekt.

Direct statement
Tijdens het laden van een ASCII-file is een direct commando gezien.

Disk error
Tijdens het formateren van een diskette is een lees- of schrijffout ontdekt.

Disk full
Er zijn geen lege sectoren meer op de diskette. Het programma kan worden vervolgd door op de RETURN-toets te drukken, doch de data is nog niet naar diskette geschreven.

Disk I/O error
Tijdens het transporteren van gegevens van of naar de disk zijn fouten geconstateerd, die het transport verhinderen.

Disk off line
De disk-drive is niet aangeschakeld.

Disk write protected
De diskette kan niet worden beschreven doordat de "write protect tab" in de stand "write protect" staat.

Division by zero
In een door de computer uit te voeren berekening blijkt dat er een getal door nul moet worden gedeeld. Dit zou een oneindig groot resultaat geven, hetgeen niet door de computer kan worden verwerkt.

Field overflow
Met het FIELD statement worden meer dan 256 bytes toegekend, of het einde van het FIELD-buffer is bereikt sequential I/O naar een random file.

File already exists
De nieuwe bestandsnaam in het NAME-statement is de naam van een reeds bestaand bestand.

File already open
De file die met het OPEN-statement wordt getracht te openen blijkt al geopend te zijn.

File not found
Het bestand dat in een LOAD, KILL of OPEN statement wordt aangeduid blijkt niet op diskette te staan.

File not open
Het bestand waar naartoe wordt geschreven, of waarvan wordt gelezen is nog niet geopend.

File still open
Het bestand is nog niet afgesloten (met CLOSE of END).

Illegal direct
In de directe mode is getracht een statement uit te voeren dat niet in die mode mag worden uitgevoerd.

Illegal function call
De aangeroepen functie is op de verkeerde manier aangeroepen. Voorbeelden hiervan zijn:
- Een negatieve of te grote subscript.
- Bij de LOG-functie werd de parameter 0 of een negatieve parameter opgegeven.
- Bij de SQR-functie werd een negatieve parameter opgegeven.
- Bij MID$, LEFT$, RIGHT$, INP, OUT, PEEK, POKE, TAB, SPC, STRING$, SPACE$, INSTR$, ON-GOTO of ON-GOSUB werd een verkeerde parameter opgegeven.

Input past end
Er wordt getracht, voorbij het einde van een bestand, nog data uit dat bestand te lezen. Dit probleem kan worden voorkomen door de EOF-functie in het programma te gebruiken.

Internal error
Er is een fout opgetreden die niet door het systeem kan worden opgelost. Er is mogelijk sprake van een hardware fout of van een fout in het MSX-software systeem.

Line buffer overflow
Er zijn te veel karakters ingetikt (>255).

Missing operand
Een statement, commando of functie heeft geen parameter die het wel zou moeten hebben.

NEXT without FOR
Er moet een NEXT-statement worden uitgevoerd, echter, er is geen (bijbehorende) FOR-statement uitgevoerd.

No RESUME
Een foutafhandelings-subroutine is gestart, doch wordt niet afgesloten met een RESUME-statement.

Out of DATA
Bij het uitvoeren van een READ-statement wijst de "pointer" voorbij het laatste DATA-element van de laatste DATA-regel. Ofwel wordt de READ-statement te vaak uitgevoerd, of het RESTORE-statement is vergeten.

Out of memory
Het programma is te groot, of bevat te veel FOR-lussen of GOSUB-statements, waardoor er "stack"-ruimte moet worden gecreeerd, terwijl daar geen ruimte meer voor is. Ook een te gecompliceerde expressie kan dit veroorzaken.

Out of string space
De voor de variabelen beschikbare geheugenruimte kan niet alle variabelen bevatten. Deze geheugenruimte kan met het CLEAR-statement worden vergroot.

Overflow
Het resultaat van een berekening is groter dan kan worden verwerkt.

Redimensioned array
Een reeds bestaande variabele, of een reeds gedimensioneerde array, mag niet (opnieuw) gedimensioneerd worden.

Rename across disk
Zowel de oude als de nieuwe bestandsnaam moet slaan op bestanden die op dezelfde disk staan. De disk mag tijdens het uitvoeren van het NAME-statement niet worden verwisseld.

RESUME without error
De computer komt, in het programma, een RESUME-statement tegen, zonder dat er een ON ERROR GOTO was uitgevoerd.

RETURN without GOSUB
De computer komt, in het programma, een RETURN-statement tegen, zonder dat er een GOSUB-statement was uitgevoerd.

Sequential I/O only
Er wordt getracht om met behulp van de GET en PUT statements, die alleen mogen worden gebruikt op random toegankelijke bestanden, gegevens uit een sequentieel bestand te lezen of er naar toe te schrijven.

String formula too complex
De string-expressie is te lang of te complex, en moet derhalve in kleinere (eenvoudiger) stukken worden opgedeeld.

String too long
Het aantal karakters dat aan een string mag worden toegekend is maximaal 255.

Subscript out of range
De subscript waarnaar wordt gerefereerd, is kleiner of groter dan de bij de array opgegeven subscripts.

Syntax error
Er is een schrijffout gemaakt in een statement, commando of functie. Check de syntaxbeschrijving van het MSX-BASIC.

Too many files
Er mogen maximaal 112 files tegelijk aanwezig zijn.

Type mismatch
Aan een numerieke variabele zijn alfanumerieke karakters toegekend, of aan een functie, die een numerieke parameter verwacht is een alfanumerieke parameter gekoppeld.

Undefined line number
Het opgegeven regelnummer komt niet in het programma voor.

Undefined user function
Er werd gerefereerd naar een user functie die niet bestaat, of die niet met DEF is gedefinieerd.

Unprintable errors
Een aantal codes hebben geen bijbehorende boodschap. In plaats daarvan wordt dan de boodschap "unprintable errors" afgedrukt.

Verify error
De gegevens die met het commando "CLOAD?" van cassette zijn gelezen, zijn niet gelijk aan de in het geheugen staande gegevens. Dit houdt in dat de gegevens op de cassette niet correct zijn. (Opnieuw wegschrijven met SAVE.)

10 Editing

De MSX-computer is voorzien van en full-screen editor. Dit wil zeggen dat alle op het scherm staande programma regels naar believen kunnen worden gewijzigd. Hiertoe heeft de gebruiker een aantal toetsen tot zijn beschikking. Het aantal mogelijkheden is zo groot, dat het niet doenlijk was voor iedere mogelijkheid een aparte functietoets op het toetsenbord aan te brengen. U zult dan ook voor een aantal mogelijkheden twee toetsen tegelijk dienen in te drukken, de CONTROL toets plus een lettertoets.

In de hierna volgende tabel zijn alle mogelijkheden opgenomen. Eerst de functies waarvoor aparte toetsen aanwezig zijn, daarna de gecombineerde aanslagen.

toets	functie
BS	Back Space. Hiermee wordt het laatst ingetikte karakter gewist.
CLS	CLear Screen. Door indrukken van de CLS/HOME-toets samen met de SHIFT-toets wordt het beeldscherm schoongemaakt.
DEL	DELete. Hiermee wordt het karakter dat onder de cursor staat gewist en schuift de tekst rechts van de cursor 1 plaatsje naar links.
HOME	Hiermee wordt de cursor naar de eerste beeldschermpositie verplaatst (linksbovenaan).
INS	INSert. Na het intikken van deze toets kunnen karakters in de tekst worden tussengevoegd. Deze karakters zullen dan op de plaats van de cursor terecht komen, terwijl alle karakters vanaf de cursor naar rechts zullen schuiven. De INS-mode wordt verlaten door het activeren van een andere functie.
Pijltjes	De pijltjestoetsen dienen voor het verplaatsen van de cursor. De cursor kan naar iedere gewenste plaats van het beeldscherm worden gebracht (in de richting van de pijltjes).

toets	functie
RETURN	Hiermee worden de op het scherm aangebrachte wijzigingen in het programmageheugen opgenomen
TAB	Hiermee wordt de cursor naar de volgende kolom verplaatst. Het beeldscherm is opgedeeld in kolommen van ieder 16 karakters breed.

Nu volgt een lijst van toetsen die tesamen met de CONTROL-toets (CTRL) moeten worden ingedrukt.

toets	functie
B	Hiermee wordt de cursor naar het begin van het vorige woord verplaatst.
C	Hiermee wordt de ingave onderbroken. De ingetikte wijzigingen worden niet in het programmageheugen opgenomen.
E	Hiermee wordt alle tekst, op de zelfde programmaregel,die de cursor volgt, gewist.
F	Hiermee wordt de cursor naar het begin van het volgende woord verplaatst.
J	Hiermee wordt de cursor naar de volgende regel verplaatst. Mocht de cursor al op de laatste regel van het beeldscherm staan, dan zal de tekst een regel omhoog "scrollen".
N	Hiermee wordt de cursor naar het einde van de programmaregel verplaatst.
U	Hiermee wordt de hele programmaregel gewist.

11 BASIC-tokens

BASIC-statements, -functies en -operators worden in het geheugen opgeslagen als codes. Elk BASIC-statement, ongeacht het aantal letters, waaruit het woord van dat statement bestaat, wordt weergegeven door een code die ligt tussen 128 en 256. Hierdoor zal een statement in het geheugen nooit meer dan 1 positie innemen. Voor functies geldt, dat de codes van de functies (128-256) worden voorafgegaan door de code 255.

In de hiernavolgende tabellen worden de statements, functies en operators en hun vervangende codes (tokens) eerst in alfabetische volgorde van statement en daarna in oplopende volgorde van code (token) afgedrukt.

statement /functie	code dec		code hex		statement /functie	code dec		code hex	
*		243		F3	CLOSE		180		B4
+		241		F1	CLS		159		9F
-		242		F2	COLOR		189		BD
/		244		F4	CONT		153		99
<		240		F0	COPY		214		D6
=		239		EF	COS	255	140	FF	8C
>		238		EE	CSAVE		154		9A
ABS	255	134	FF	86	CSNG	255	159	FF	9F
AND		246		F6	CSRLIN		232		E8
ASC	255	149	FF	95	CVD	255	170	FF	AA
ATN	255	142	FF	8E	CVI	255	168	FF	A8
AUTO		169		A9	CVS	255	169	FF	A9
BASE		201		C9	DATA		132		84
BEEP		192		C0	DEF		151		97
BIN$	255	157	FF	9D	DEFDBL		174		AE
BLOAD		207		CF	DEFINT		172		AC
BSAVE		208		D0	DEFSNG		173		AD
CALL		202		CA	DEFSTR		171		AB
CDBL	255	160	FF	A0	DELETE		168		A8
CHR$	255	150	FF	96	DIM		134		86
CINT	255	158	FF	9E	DRAW		190		BE
CIRCLE		188		BC	DSKF	255	166	FF	A6
CLEAR		146		92	END		129		81
CLOAD		155		9B	EOF	255	171	FF	AB

statement /functie	code dec		code hex	
ERASE		165		A5
ERL		225		E1
ERR		226		E2
ERROR		166		A6
EXP	255	139	FF	8B
FIELD		177		B1
FILES		183		B7
FIX	255	161	FF	A1
FN		222		DE
FOR		130		82
FRE	255	143	FF	8F
GET		178		B2
GOSUB		141		8D
GOTO		137		89
HEX$	255	155	FF	9B
IF		139		8B
INKEY$		236		EC
INP	255	144	FF	90
INPUT		133		85
INSTR		229		E5
INT	255	133	FF	85
KEY		204		CC
KILL		212		D4
LEFT$	255	129	FF	81
LEN	255	146	FF	92
LET		136		88
LFILES		187		BB
LINE		175		AF
LIST		147		93
LLIST		158		9E
LOAD		181		B5
LOC	255	172	FF	AC
LOCATE		216		D8
LOF	255	173	FF	AD
LOG	255	138	FF	8A
LPOS	255	156	FF	9C
LPRINT		157		9D
LSET		184		B8
MAX		205		CD
MERGE		182		B6
MID$	255	131	FF	83
MKD$	255	176	FF	B0
MKI$	255	174	FF	AE
MKS$	255	175	FF	AF
MOTOR		206		CE
NAME		211		D3
NEW		148		94
NEXT		131		83
NOT		224		E0
OCT$	255	154	FF	9A
OFF		235		EB
ON		149		95
OPEN		176		B0
OR		247		F7
OUT		156		9C
PAD	255	165	FF	A5
PAINT		191		BF
PDL	255	164	FF	A4
PEEK	255	151	FF	97
PLAY		193		C1
POINT		237		ED
POKE		152		98
POS	255	145	FF	91
PRESET		195		C3
PRINT		145		91
PSET		194		C2
PUT		179		B3
READ		135		87
REM		143		8F
RENUM		170		AA
RESTORE		140		8C
RESUME		167		A7
RETURN		142		8E
RIGHT$	255	130	FF	82
RND	255	136	FF	88
RSET		185		B9
RUN		138		8A
SAVE		186		BA
SCREEN		197		C5
SET		210		D2
SGN	255	132	FF	84
SIN	255	137	FF	89
SOUND		196		C4
SPACE$	255	153	FF	99

statement /functie	code dec		hex	
SPC(		223		DF
SPRITE		199		C7
SQR	255	135	FF	87
STEP		220		DC
STICK	255	162	FF	A2
STOP		144		90
STR$	255	147	FF	93
STRIG	255	163	FF	A3
STRING$		227		E3
SWAP		164		A4
TAB(		219		DB
TAN	255	141	FF	8D
TIME		203		CB
TO		217		D9
TROFF		163		A3
TRON		162		A2
USING		228		E4
USR		221		DD
VAL	255	148	FF	94
VARPTR		231		E7
VDP		200		C8
VPEEK	255	152	FF	98
VPOKE		198		C6
WAIT		150		96
WIDTH		160		A0
XOR		248		F8
\		252		FC
^		245		F5

code dec	hex	statement /functie
129	81	END
130	82	FOR
131	83	NEXT
132	84	DATA
133	85	INPUT
134	86	DIM
135	87	READ
136	88	LET
137	89	GOTO
138	8A	RUN
139	8B	IF
140	8C	RESTORE
141	8D	GOSUB
142	8E	RETURN
143	8F	REM
144	90	STOP
145	91	PRINT
146	92	CLEAR
147	93	LIST
148	94	NEW
149	95	ON
150	96	WAIT
151	97	DEF
152	98	POKE
153	99	CONT
154	9A	CSAVE
155	9B	CLOAD
156	9C	OUT
157	9D	LPRINT
158	9E	LLIST
159	9F	CLS
160	A0	WIDTH
162	A2	TRON
163	A3	TROFF
164	A4	SWAP
165	A5	ERASE
166	A6	ERROR
167	A7	RESUME
168	A8	DELETE
169	A9	AUTO
170	AA	RENUM
171	AB	DEFSTR
172	AC	DEFINT
173	AD	DEFSNG

code		statement	code		statement
dec	hex	/functie	dec	hex	/functie
174	AE	DEFDBL	220	DC	STEP
175	AF	LINE	221	DD	USR
176	B0	OPEN	222	DE	FN
177	B1	FIELD	223	DF	SPC(
178	B2	GET	224	E0	NOT
179	B3	PUT	225	E1	ERL
180	B4	CLOSE	226	E2	ERR
181	B5	LOAD	227	E3	STRING$
182	B6	MERGE	228	E4	USING
183	B7	FILES	229	E5	INSTR
184	B8	LSET	231	E7	VARPTR
185	B9	RSET	232	E8	CSRLIN
186	BA	SAVE	235	EB	OFF
187	BB	LFILES	236	EC	INKEY$
188	BC	CIRCLE	237	ED	POINT
189	BD	COLOR	238	EE	>
190	BE	DRAW	239	EF	=
191	BF	PAINT	240	F0	<
192	C0	BEEP	241	F1	+
193	C1	PLAY	242	F2	-
194	C2	PSET	243	F3	*
195	C3	PRESET	244	F4	/
196	C4	SOUND	245	F5	^
197	C5	SCREEN	246	F6	AND
198	C6	VPOKE	247	F7	OR
199	C7	SPRITE	248	F8	XOR
200	C8	VDP	252	FC	\
201	C9	BASE	255 129	FF 81	LEFT$
202	CA	CALL	255 130	FF 82	RIGHT$
203	CB	TIME	255 131	FF 83	MID$
204	CC	KEY	255 132	FF 84	SGN
205	CD	MAX	255 133	FF 85	INT
206	CE	MOTOR	255 134	FF 86	ABS
207	CF	BLOAD	255 135	FF 87	SQR
208	D0	BSAVE	255 136	FF 88	RND
210	D2	SET	255 137	FF 89	SIN
211	D3	NAME	255 138	FF 8A	LOG
212	D4	KILL	255 139	FF 8B	EXP
214	D6	COPY	255 140	FF 8C	COS
216	D8	LOCATE	255 141	FF 8D	TAN
217	D9	TO	255 142	FF 8E	ATN
219	DB	TAB(	255 143	FF 8F	FRE

code dec		hex		statement /functie	code dec		hex		statement /functie
255	144	FF	90	INP	255	160	FF	A0	CDBL
255	145	FF	91	POS	255	161	FF	A1	FIX
255	146	FF	92	LEN	255	162	FF	A2	STICK
255	147	FF	93	STR$	255	163	FF	A3	STRIG
255	148	FF	94	VAL	255	164	FF	A4	PDL
255	149	FF	95	ASC	255	165	FF	A5	PAD
255	150	FF	96	CHR$	255	166	FF	A6	DSKF
255	151	FF	97	PEEK	255	168	FF	A8	CVI
255	152	FF	98	VPEEK	255	169	FF	A9	CVS
255	153	FF	99	SPACE$	255	170	FF	AA	CVD
255	154	FF	9A	OCT$	255	171	FF	AB	EOF
255	155	FF	9B	HEX$	255	172	FF	AC	LOC
255	156	FF	9C	LPOS	255	173	FF	AD	LOF
255	157	FF	9D	BIN$	255	174	FF	AE	MKI$
255	158	FF	9E	CINT	255	175	FF	AF	MKS$
255	159	FF	9F	CSNG	255	176	FF	B0	MKD$

12 MSX-karakterset

				BIT 4,5,6,7																
					0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
					0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
					0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
					0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
BIT 0,1,2,3				HEX 1 / 2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	Null		Spatie	θ	@	P	`	p	Ç	É	á	Ã			α	≡
0	0	0	1	1	☺		!	1	A	Q	a	q	ü	æ	í	ã			β	±
0	0	1	0	2	☻		"	2	B	R	b	r	é	Æ	ó	Ĩ			γ	≥
0	0	1	1	3	♥		#	3	C	S	c	s	â	ô	ú	ĩ			π	≤
0	1	0	0	4	♦		$	4	D	T	d	t	ä	ö	ñ	Õ			Σ	⌠
0	1	0	1	5	♣		%	5	E	U	e	u	à	ò	Ñ	õ			σ	⌡
0	1	1	0	6	♠		&	6	F	V	f	v	å	û	ª	Ũ			μ	÷
0	1	1	1	7	•		'	7	G	W	g	w	ç	ù	º	ũ			τ	≈
1	0	0	0	8	◘		(	8	H	X	h	x	ê	ÿ	¿	Ĳ		▽	φ	°
1	0	0	1	9	○		)	9	I	Y	i	y	ë	Ö	⌐	ĳ		‡	θ	•
1	0	1	0	A	◙		*	:	J	Z	j	z	è	Ü	¬	¾		₩	Ω	•
1	0	1	1	B	♂		+	;	K	[	k	{	ï	¢	½	~			δ	√
1	1	0	0	C	♀		,	<	L	\	l	¦	î	£	¼	◊			∞	η
1	1	0	1	D	♪		–	=	M	]	m	}	ì	¥	¡	‰			φ	²
1	1	1	0	E	♫		.	>	N	∧	n	~	Ä	Pt	«	¶			∈	■
1	1	1	1	F	☼		/	?	O	_	o	△	Å	ƒ	»	§			∩	Trans-parant

13 ASCII-karakterset

De meest gebruikte Internationale gestandaardiseerde code voor karakters is de ASCII-code. ASCII is de afkorting van American Standard Code for Information Interchange. De hiernavolgende tabel kan als volgt worden gebruikt:

Opzoeken welke code bij een gegeven karakter hoort.

Stel u zoekt de code die bij de letter S behoort. Zoek dan de letter S op in de tabel. Kijk nu welk getal er boven de betreffende kolom staat. Dit is het meest significante deel van de code. Kijk nu welk getal er links van de betreffende kolom staat. Dit is het minst significante deel van de code. Voor de letter S vindt u zodoende de code Hex. 53, of Bin. 0101 0011.

Opzoeken welke letter bij een gegeven code hoort.

Stel u hebt de code hex. 54. Zoek eerst de kolom waarboven de waarde 5 staat (langs de bovenrand). Ga nu zover naar beneden tot u de rij hebt bereikt waarnaast (aan de linker kant) de waarde hex. 4 staat. U hebt dan het vak bereikt waarin de gezochte letter staat. In dit vak vindt u de letter T.

binair		0000	0001	0010	0011	0100	0101	0110	0111
	hex	0	1	2	3	4	5	6	7
0000	0	NUL	DLE	SP	0	@	P	`	p
0001	1	SOH	DC1	!	1	A	Q	a	q
0010	2	STX	DC2	"	2	B	R	b	r
0011	3	ETX	DC3	#	3	C	S	c	s
0100	4	EOT	DC4	$	4	D	T	d	t
0101	5	ENQ	NAK	%	5	E	U	e	u
0110	6	ACK	SYN	&	6	F	V	f	v
0111	7	BEL	ETB	'	7	G	W	g	w
1000	8	BS	CAN	(	8	H	X	h	x
1001	9	HT	EM	)	9	I	Y	i	y
1010	A	LF	SUB	*	:	J	Z	j	z
1011	B	VT	ESC	+	;	K	[	k	{
1100	C	FF	FS	,	<	L	/	l	|
1101	D	CR	GS	-	=	M	]	m	}
1110	E	SO	RS	.	>	N	^	n	
1111	F	SI	US	/	?	O	_	o	DEL

14 Codes van functietoetsen

Codes van een aantal veelgebruikte toetsen:

toets	code dec	code hex
BS	8	08
CLS	12	0C
DEL	127	7F
ESC	27	1B
HOME	11	0B
INS	18	12
RETURN	13	0D
SELECT	24	18
TAB	9	09

toets	code dec	code hex
↑	30	1E
→	28	1C
↓	31	1F
←	29	1D

Het volgende routinetje geeft de mogelijkheid om voor iedere ingedrukte toets te zien welke code daarbij hoort en welk karakter dat tot gevolg heeft:

```
10 I$=INKEY$
20 IF I$<>"" THEN PRINT ASC(I$),I$
30 GOTO 10
```

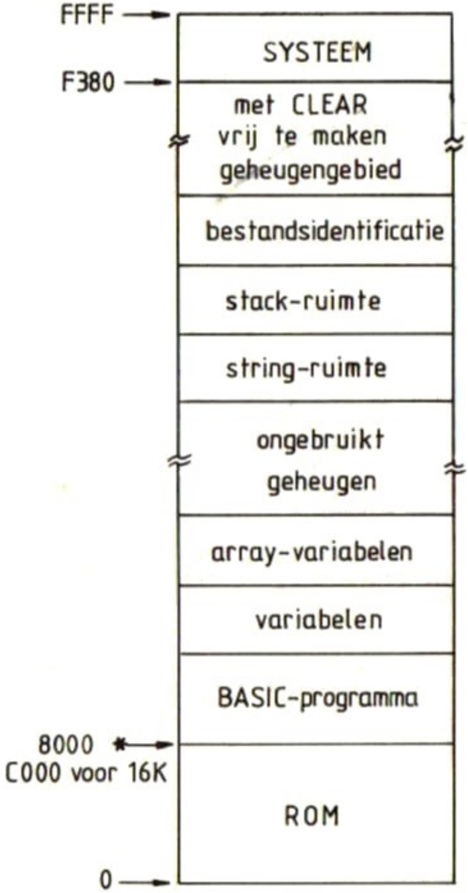
15 Scannen van toetsen

De toetsen van het toetsenbord zijn in een matrix opgenomen. Wanneer de software het toetsenbord wil lezen, dan wordt een rij (Y0 tot en met Y8) uit de matrix geselecteerd, waarna de status van de op die rij voorkomende toetsen wordt gelezen. Is er geen enkele toets ingedrukt, dan zijn alle bits (X7 tot en met X0) in de gelezen rij "hoog" ofwel 1. De ingedrukte toets zal een "laag" ofwel 0 tot gevolg hebben. (zie onderstaande matrix)

Met behulp van het BIOS-entry-point SNSMAT (&H0141) kan direct van deze matrix gebruik worden gemaakt. Plaats daartoe het rijnummer in de accumulator. Het resultaat zal zijn, dat de accumulator, door middel van nul(len) in de overeenkomstige bitposities, de ingedrukte toets(en) weergeeft.

kolom / rij	X7	X6	X5	X4	X3	X2	X1	X0
Y0	7	6	5	4	3	2	1	0
Y1	;	[	@	\	^	-	9	8
Y2	B	A	_	/	.	,	]	:
Y3	J	I	H	G	F	E	D	C
Y4	R	Q	P	O	N	M	L	K
Y5	Z	Y	X	W	V	U	T	S
Y6	F3	F2	F1	CODE	CAP	GRAPH	CTRL	SHIFT
Y7	RET	SEL	BS	STOP	TAB	ESC	F5	F4
Y8	->	v	^	<-	DEL	INS	HOME	SPACE

16 Geheugen lay-out



* 32K ROM + 16K RAM dan hier een niet gebruikte ruimte van 16K.

17 MSXDOS commando's

In de navolgende beschrijving kan overal, waar een bestandsnaam of bestandsnaamuitbreiding wordt genoemd, in deze naam gebruik worden gemaakt van vervangende tekens (* en ?).

- * Vervangt 1 of meer tekens.
- ? Vervangt 1 teken.

Voorbeelden:

*.BAS wil zeggen:
Alle bestandsnamen met de bestandsnaamuitbreiding BAS.

B*.* wil zeggen:
Alle bestandsnamen die met een letter B beginnen.

. wil zeggen:
Alle bestandsnamen.

TEST? wil zeggen:
Alle bestandsnamen van 5 tekens, waarvan de eerste vier tekens de letters TEST zijn en die geen bestandsnaamuitbreiding hebben.

T??.* wil zeggen:
Alle bestandsnamen van drie tekens, waarvan het eerste teken de letter T is.

Naast de hierna te beschrijven commando's zijn er ook een aantal control-sequences, waarmee u invloed kunt uitoefenen op de werking van de commando's:

CONTROL + P

Hiermee wordt alle informatie die naar het beeldscherm wordt gestuurd tevens naar de printer gestuurd. Zorg er wel voor dat de printer aangesloten en aan staat.

CONTROL + N

Hiermee wordt het CONTROL+P commando ongedaan gemaakt.

CONTROL + S

Hiermee wordt het in uitvoering zijnde commando tijdelijk onderbroken. Door een willekeurige toets in te drukken wordt de uitvoering van het commando voortgezet.

CONTROL + C

Hiermee wordt het in uitvoering zijnde commando afgebroken.

Om het intoetsen van commando's te versnellen wordt het commando opgeslagen in een klein stukje geheugen, dat template wordt genoemd. Om de gegevens weer uit de template te kunnen lezen, kan van een aantal toetsen gebruik worden gemaakt. Het is mogelijk om nieuw ingetikte tekens te combineren met tekens uit de template. De template-controle toetsen zijn:

CURSOR RECHTS

Copieert het volgende teken uit de template naar het invoerbuffer.

CURSOR NAAR BENEDEN

Copieert alle resterende tekens uit de template naar het invoerbuffer.

CURSOR LINKS

Wist het laatste teken uit het invoerbuffer (= BS-toets).

CURSOR OMHOOG

Wist het gehele invoerbuffer, doch laat de inhoud van de template intact.

SELECT<teken>

Door het indrukken van de SELECT-toets, gevolgd door een letter, worden alle tekens uit de template tot en met de opgegeven letter naar het invoerbuffer gecopieerd.

DEL

Het volgende teken uit de template wordt niet naar het invoerbuffer gecopieerd.

CLS<teken>

Door het indrukken van de SHIFT + CLS toets, gevolgd door een letter, worden alle tekens uit de template tot de opgegeven letter overgeslagen (niet gecopieerd naar het invoerbuffer).

INS

Deze toets is de aan/uit-schakelaar van de INSERT-mode. Indien de INSERT-mode aan staat, kunnen 1 of meer tekens worden tussengevoegd. Staat de INSERT-mode uit, dan worden tekens overschreven.

HOME

Copieert het invoerbuffer naar de template.

Dan volgt nu een opsomming van de verschillende MSXDOS commando's, de syntax en de functie daarvan:

BASIC [<dk>:][<prognaam>[.<ext>]]

Hiermee wordt vanuit MSXDOS naar BASIC overgeschakeld. Door achter dit commando een programmanaam te vermelden, zal dat programma onmiddellijk na het overschakelen worden geladen en gestart.
Terugkeer vanuit BASIC naar MSXDOS is mogelijk met het BASIC-commando CALL SYSTEM. De actieve schijf moet dan wel de MSXDOS-bestanden bevatten.

<dk> = schijfaanduiding (A t/m F).
<prognaam>= bestandsnaam van het te starten BASIC-programma (max. 8 tekens).
<ext> = bestandsnaam-uitbreiding (max. 3 tekens).

COPY [<dev>][<file>] [<dev>][<file>]

Copieert het bestand van het eerstgenoemde apparaat naar het bestand op het laatstgenoemde apparaat. Indien <dev> wordt weggelaten, wordt de actieve schijf genomen. Indien een ander apparaat dan een schijveneenheid wordt aangegeven, behoeft geen bestandsnaam te worden opgegeven.
Het is niet noodzakelijk voor de bron en de bestemming hetzelfde type apparaat te kiezen. Het is bijvoorbeeld ook toegestaan om van toetsenbord

naar schijf of van schijf naar printer of van toetsenbord naar printer te copieren. Voorbeelden:

```
COPY CON A:FILE1
COPY A:FILE1 PRN
COPY CON PRN
```

<dev> = A t/m F - (schijveneenheid).
CON - (toetsenbord).
PRN - (printer).
AUX - (RS232-poort).
NUL - (dummy-apparaat).

<file> = Bestandsnaam (max. 8 tekens), eventueel met uitbreiding (max. 3 tekens), van elkaar gescheiden door een punt.

Aan de bestandsnamen (<file>) kan nog een parameter worden toegevoegd, namelijk /A of /B.

/A - Geeft aan dat het betreffende bestand een ASCII-bestand is.

/B - Geeft aan dat het betreffende bestand een binair bestand is.

Voorbeeld: COPY file1/A file2/B

DATE [<datum>]

Hiermee kunt u een datum aan MSXDOS opgeven. MSXDOS gebruikt deze datum in de bestandsadministratie (zie het commando DIR)
Door geen datumaanduiding op te geven zal MSXDOS u informeren over de actuele MSXDOS-datum, waarna u wordt gevraagd alsnog een nieuwe datum in te geven.

<datum> = MM.DD.JJ[JJ] (een van de hiernaast genoemde formaten.)
MM-DD-JJ[JJ]
MM/DD/JJ[JJ]

DEL [<dk>:][<file>[.<ext>]]

Het commando DEL werkt precies als het commando ERASE. DEL is alleen in MSXDOS opgenomen vanwege compatibiliteit met het BASIC-commando DELETE. Zie verder de beschrijving van ERASE.

DIR [<dk>:][<file>[.<ext>]][/P][/W]

Drukt een inhoudsopgave van de aangegeven schijf (<dk>) af. Indien een bestandsnaam is opgegeven, zullen alleen bestanden met de aangegeven naam in die inhoudsopgave worden opgenomen.
Van ieder bestand worden achtereenvolgens de naam, de grootte en de creatie-datum in de inhoudsopgave afgedrukt. Op sommige systemen wordt bovendien de creatie-tijd afgedrukt.
Indien geen schijf wordt aangegeven zal een inhoudsopgave van de actieve schijf worden gemaakt.
De inhoudsopgave wordt afgesloten met een opgave van het totaal aantal op de schijf staande files en de nog beschikbare vrije ruimte op die schijf.

- <dk> - A t/m F (schijveneenheid).
- <file> - bestandsnaam (max. 8 tekens).
- <ext> - uitbreiding (max. 3 tekens).
- /P - Door deze parameter toe te voegen stopt het afdrukken van de inhoudsopgave op het moment dat het scherm vol is. Het afdrukken wordt voortgezet nadat een willekeurige toets is ingedrukt.
- /W - Door deze parameter toe te voegen wordt een verkorte inhoudsopgave afgedrukt. In deze inhoudsopgave staan alleen nog de bestandsnamen, terwijl grootte en creatiedatum zijn weggelaten. Zo kunnen er meerdere bestandsnamen op een regel worden afgedrukt.

ERASE [<dk>:]<file>[.<ext>]

Verwijdert het aangegeven bestand van de aangegeven schijf. Dit commando komt overeen met het BASIC-commando KILL. In plaats van ERASE mag ook het commando DEL worden gebruikt.

- <dk> - A t/m F (schijveneenheid).
- <file> - bestandsnaam (max. 8 tekens).
- <ext> - uitbreiding (max. 3 tekens).

Door in plaats van een bestandsnaam een * te gebruiken kunnen alle bestanden van de schijf worden gewist. Voorbeelden:

ERASE *.* - wist alle bestanden van de actuele schijf.

ERASE *.BAS - wist alle bestanden met de uitbreiding BAS van de actuele schijf.

FORMAT

Formateert een schijfje. Voordat met formateren wordt begonnen, wordt u eerst gevraagd in welke schijveneenheid de te formateren schijf ligt. Formateren wil zeggen: indelen van de schijf in sporen (tracks) en sectoren. MSXDOS kent een groot aantal formaten, waarvan de belangrijkste in onderstaande tabel zijn opgenomen.

disk-type	aantal sides	tracks /side	sectors /track	bytes /sector	capa-citeit
1	1	80	9	512	360k
2	2	80	9	512	720k
3	1	80	8	512	320k
4	2	80	8	512	640k
5	1	40	9	512	180k
6	2	40	9	512	360k
7	1	40	8	512	160k
8	2	40	8	512	320k

Hierbij kan worden opgemerkt dat de formaten 1 en 2 meestal worden gebruikt voor 3,5 inch floppies, terwijl formaten 5 en 6 de meest voorkomende 5,25 inch floppies zijn. Deze laatste floppies zijn compatibel met floppies van Personal Computers. Welk van voornoemde formaten uw floppies krijgen, ligt vast in de bij uw schijf geleverde ROM-software.

MODE <regellengte>

Stelt het scherm in op het opgegeven aantal tekens. Dit commando komt overeen met het BASIC-commando WIDTH.

1 =< regellengte =< 32 - SCREEN 1
33 < regellengte =< 40 - SCREEN 0 (40 tekens)
regellengte = 80 - SCREEN 0 (80 tekens)

PAUSE [<commentaar>]

Dit commando drukt het <commentaar> af op het beeldscherm en vraagt vervolgens "Strike any key when ready ...". Na het indrukken van een willekeurige toets wordt de uitvoering van de commando-reeks voortgezet.

REM [<commentaar>]

Dit commando drukt het <commentaar> af op het beeldscherm en gaat daarna door met het uitvoeren van de commando-reeks.

REN[AME] [<dk>:][<file>] [<dk>:][<file>]

Met dit commando kan een nieuwe naam aan een reeds bestaande file worden gegeven. Het commando komt overeen met het BASIC-commando NAME ... TO ...
Ook in dit commando kan weer gebruik worden gemaakt van de tekens * en ?, waardoor er meerdere bestanden tegelijkertijd een nieuwe naam kunnen krijgen.

<dk> - A t/m F (schijveneenheid).
<file> - bestandsnaam (max. 8 tekens).
<ext> - uitbreiding (max. 3 tekens).

TIME [<tijd>][P]

Hiermee kunt u een tijd aan MSXDOS opgeven. MSXDOS gebruikt deze tijd in de bestandsadministratie (zie het commando DIR)
Door de tijd onmiddellijk te laten volgen door de letter P, geeft u aan dat de opgegeven tijd na 12 uur 's middags ligt. (P=Post=namiddag).
Door geen tijdsaanduiding op te geven zal MSXDOS u informeren over de actuele MSXDOS-tijd, waarna u wordt gevraagd alsnog een nieuwe tijd in te geven.

<tijd> =	UU	UUP
	UU:MM	UU:MMP
	UU:MM:SS	UU:MM:SSP

TYPE [<dk>:]<file>[.<ext>]

Drukt het aangegeven bestand af op het beeldscherm. Indien het aangegeven bestand geen ASCII-bestand is, zal de beeldscherm-output vreemde vormen kunnen aannemen.
<dk> - A t/m F (schijveneenheid).
<file> - bestandsnaam (max. 8 tekens).
<ext> - uitbreiding (max. 3 tekens).

VERIFY [ON][OFF]

Nadat het commando VERIFY ON is gegeven, zal elk bestand, nadat het naar schijf is geschreven, worden teruggelezen en gecontroleerd op correctheid. Dit commando blijft van kracht, totdat het commando VERIFY OFF is gegeven.

Met behulp van de hiervoor omschreven commando's kunnen nieuwe commando's worden samengesteld, door een combinatie van de omschreven commando's in een zogenaamde BAT-file te zetten. Door vervolgens op de MSXDOS-prompt de naam van die BAT-file in te tikken, zullen alle commando's uit de BAT-file achtereenvolgens worden uitgevoerd.
Een bijzondere vorm van BAT-file is de BAT-file met de naam AUTOEXEC. Na het opstarten van MSXDOS (direct na het aanschakelen van het systeem) wordt op de default-schijf gekeken of daar een bestand met de naam AUTOEXEC.BAT staat. Is deze file aanwezig, dan worden de commando's daaruit automatisch uitgevoerd.
Omdat de commando's in BAT-files vastliggen, is er een mogelijkheid geschapen bepaalde parameters variabel te maken. Hiertoe wordt in de commando-regel in plaats van een naam een variabele gebruikt. De toegestane variabelen zijn %0 tot en met %9.

Aanroepen van een .BAT-file:

```
<naam> <par1> . . . <par9>
```

De variabelen in de .BAT-file worden vervangen door de achter <naam> gegeven parameters <par1> etc.:

```
%1 - eerste parameter
%2 - tweede parameter
%3 - derde parameter
etc.
%0 krijgt de waarde van <naam>
```

Voorbeeld:

Creeren van een zelfgemaakt commando:

```
COPY CON INH.BAT
DIR %1:
^Z
```

Aanroepen van het zelfgemaakte commando:

```
INH B
```

De letter B is hierin de parameter, die wordt toegekend aan de variabele %1. Er zal dus een inhoudsopgave worden gemaakt van schijf B.

18 VT52 escape sequences in MSX

De beeldschermbesturing van de MSX is tot op zekere hoogte compatibel met de standaard VT52 terminal. De door MSX ondersteunde VT52 escape sequences zijn:

```
ESC A = CHR$(27);"A"   Cursor omhoog.
ESC B = CHR$(27);"B"   Cursor naar beneden.
ESC C = CHR$(27);"C"   Cursor rechts.
ESC D = CHR$(27);"D"   Cursor links.
ESC E = CHR$(27);"E"   Clear Screen.
ESC H = CHR$(27);"H"   Home.
ESC J = CHR$(27);"J"   Wis  vanaf  cursor  tot  einde
                       scherm.
ESC K = CHR$(27);"K"   Wis  vanaf  cursor  tot  einde
                       regel.
ESC L = CHR$(27);"L"   Voeg een regel tussen.
ESC M = CHR$(27);"M"   Wis een regel.
ESC j = CHR$(27);"j"   Clear Screen.
ESC l = CHR$(27);"l"   Wis de hele regel.

ESC Y<v+32><h+32>      Cursor naar positie v,h.
                       v = verticaal, h = horizontaal.
```

Voorbeeld om de cursor naar positie 28 van regel 7 te verplaatsen:

```
    ESC Y<7+32><28+32> = ESC Y<39><60> = CHR$(27);"Y'<"
```

```
ESC x4 = CHR$(27);"x4"   Cursor is blok.
ESC y4 = CHR$(27);"y4"   Cursor is lijn.
ESC x5 = CHR$(27);"x5"   Cursor onzichtbaar.
ESC y5 = CHR$(27);"y5"   Cursor zichtbaar.
```

Het volgende programma laat een toepassingsmogelijkheid van de escape sequences zien:

19 Z80 interrupt modes

De Z80 CPU kan in een van de volgende drie interrupt modes werken:

Mode 0
Indien de CPU in deze mode is, kan het "device", dat de interrupt genereert, iedere gewenste instructie op de databus zetten en deze instructie door de CPU laten uitvoeren.

Mode 1
In deze mode zal de CPU op een interrupt reageren door automatisch een RST-instructie uit te voeren. Deze RST instructie zal een "restart" op adres 0038 (hex.) ten gevolge hebben. De oude inhoud van de programma teller PC wordt op de stack geschreven.

Mode 2
In deze mode kan een indirecte sprong naar ieder gewenst geheugenadres worden gemaakt. De CPU vormt een adres uit de inhoud van het I-register (msb) en een byte dat door het "device", dat de interrupt genereert, wordt verstrekt (lsb). Dit gevormde adres wijst naar het eerste byte van een twee bytes veld. Dat veld wijst naar een service routine. De CPU zal deze service routine automatisch starten.

20 Z80- registers

Hoofdregisterset		Hulpregisterset	
accumulator A	vlaggen F	accumulator A'	vlaggen F'
B	C	B'	C'
D	E	D'	E'
H	L	H'	L'

Interrupt vector I	Memory refresh R
Index register IX	
Index register IY	
Program counter PC	

Accumulator en vlagregisters.

De Z80 CPU bevat twee onafhankelijke 8-bits A-registers met bijbehorende vlagregisters. De accumulator bevat het resultaat van 8-bits rekenkundige of logische bewerkingen, terwijl het vlagregister specifieke condities voor 8 en 16 bits bewerkingen bevat, zoals het aangeven of het resultaat van een bewerking al of niet 0 is. De programmeur kan zelf kiezen welk paar (accumulator + vlagregister) hij wil gebruiken, door gebruik te maken van een exchange-instructie.

Algemene registers.

Er zijn twee sets van ieder zes 8-bits registers. Deze registers kunnen als afzonderlijke 8-bits registers of als drie paren van 16-bits registers worden gebruikt. De programmeur kan een van beide sets selecteren met behulp van exchange instructies.

Het I-register.

Het I-register wordt in interrupt mode 2 gebruikt om er de 8 meest significante adresbits, van het 16-bits indirecte adres, uit te lezen. De 8 minst significante adresbits worden door het "device" dat de interrupt veroorzaakt gegeven.

Het R-register.

Dit register werkt als een teller, die automatisch wordt verhoogd. De inhoud van de teller wordt op de minst significante adresbits gezet, samen met een refresh-control signaal. Normaliter wordt dit register niet door de programmeur gebruikt.

Index registers IX en IY.

Deze registers worden in de "Indexed addressing mode" gebruikt. Ze bevatten een 16-bits adres. De adressen in deze registers worden als basisadres van een geheugengebied gebruikt. Deze registers zijn vooral handig bij het behandelen van tabellen.

Stack Pointer.

Het 16-bits adres in dit register (SP) wijst naar de top van de stack. Deze stack heeft een LIFO-organisatie. Met behulp van de PUSH en POP-instructies kunnen gegevens op de stack worden gezet, of er van af worden gehaald.

Program Counter.

Het PC-register bevat het 16-bits adres van de instructie die uit het geheugen moet worden gelezen. Dit adres wordt na het inlezen van een instructie automatisch verhoogd naar het adres van de volgende instructie. Met behulp van spronginstructies kan het adres ook worden veranderd.

21 Symbolische omschrijving van Z80-instructies

Mnemonic	Symbolische omschrijving
ADC r	A <-- A+r+CY
ADC n	A <-- A+n+CY
ADC (HL)	A <-- A+(HL)+CY
ADC (IX+d)	A <-- A+(IX+d)+CY
ADC (IY+d)	A <-- A+(IY+d)+CY
ADC HL,ss	HL <-- HL+ss+CY
ADD r	A <-- A+r
ADD n	A <-- A+n
ADD (HL)	A <-- A+(HL)
ADD (IX+d)	A <-- A+(IX+d)
ADD (IY+d)	A <-- A+(IY+d)
ADD HL,ss	HL <-- HL+ss
ADD IX,pp	IX <-- IX+pp
ADD IY,rr	IY <-- IY+rr
AND r	A <-- A ∧ r
AND n	A <-- A ∧ n
AND (HL)	A <-- A ∧ (HL)
AND (IX+d)	A <-- A ∧ (IX+d)
AND (IY+d)	A <-- A ∧ (IY+d)
BIT b,r	Z <-- $\overline{r}$b
BIT b,(HL)	Z <-- $\overline{(HL)}$b
BIT b,(IX+d)	Z <-- $\overline{(IX+d)}$b
BIT b,(IY+d)	Z <-- $\overline{(IYkd)}$b
CALL cc,nn	alleen uitvoeren indien cc waar is.
	(SP-1) <-- PCH
	(SP-2) <-- PCL
	PC <-- nn
CALL nn	(SP-1) <-- PCH
	(SP-2) <-- PCL
	PC <-- nn
CCF	CY <-- $\overline{CY}$
CP r	A : r
CP n	A : n
CP (HL)	A : (HL)
CP (IX+d)	A : (IX+d)
CP (IY+d)	A : (IY+d)
CPD	A : (HL)
	HL <-- HL-1
	BC <-- BC-1
CPDR	Als CPD, doch net zolang herhaald tot A = (HL) of BC = 0.

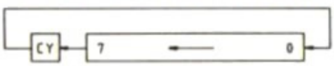
Mnemonic	Symbolische omschrijving
CPI	A : (HL) HL <-- HL+1 BC <-- BC-1
CPIR	Als CPI, doch net zolang herhaald tot A = (HL) of BC = 0.
CPL	A <-- $\overline{A}$
DAA	Converteert de inhoud van reg. A naar packed BCD.
DEC r	r <-- r-1
DEC (HL)	(HL) <-- (HL)-1
DEC (IX+d)	(IX+d) <-- (IX+d)-1
DEC (IY+d)	(IY+d) <-- (IY+d)+1
DEC ss	ss <-- ss-1
DEC IX	IX <-- IX-1
DEC IY	IY <-- IY-1
DI	IFF <-- 0
DJNZ e	B <-- B-1 Indien B=0 dan volgende instructie. Indien B<>0 dan: PC <-- PC+e
EI	IFF <-- 1
EX AF,AF'	AF <--> AF'
EX DE,HL	DE <--> HL
EX (SP),HL	H <--> (SP+1) L <--> (SP)
EX (SP),IX	IXH <--> (SP+1) IXL <--> (SP)
EX (SP),IY	IYH <--> (SP+1) IYL <--> (SP)
EXX	BC <--> BC' DE <--> DE' HL <--> HL'
HALT	CPU wordt gestopt
IM 0	Interrupt mode 0
IM 1	Interrupt mode 1
IM 2	Interrupt mode 2
IN A,(n)	A <-- (n) (n naar A0-A7, A naar A8-A15)
IN r,(C)	r <-- (C) (C naar A0-A7, B naar A8-A15)
INC r	r <-- r+1
INC (HL)	(HL) <-- (HL)+1
INC (IX+d)	(IX+d) <-- (IX+d)+1
INC (IY+d)	(IY+d) <-- (IY+d)+1
INC ss	ss <-- ss+1

Mnemonic	Symbolische omschrijving
INC IX	IX <-- IX+1
INC IY	IY <-- IY+1
IND	(HL) <-- (C) B <-- B-1 HL <-- HL-1
INDR	Als IND, totdat B=0.
INI	(HL) <-- (C) B <-- B-1 HL <-- HL+1
INIR	Als INI, totdat b=0.
JP cc,nn	Indien cc is waar, dan PC <-- nn
JP nn	PC <-- nn
JP (HL)	PC <-- HL
JP (IX)	PC <-- IX
JP (IY)	PC <-- IY
JR C,e	Indien Carry=1 dan PC <-- PC+e
JR e	PC <-- PC+e
JR NC,e	Indien Carry=0 dan PC <-- PC+e
JR NZ,e	Indien Zero=0 dan PC <-- PC+e
JR Z,e	Indien Zero=1 dan PC <-- PC+e
LD r,r'	r <-- r'
LD r,n	r <-- n
LD r,(HL)	r <-- (HL)
LD r,(IX+d)	r <-- (IX+d)
LD r,(IY+d)	r <-- (IY+d)
LD (HL),r	(HL) <-- r
LD (IX+d),r	(IX+d) <-- r
LD (IY+d),r	(IY+d) <-- r
LD (HL),n	(HL) <-- n
LD (IX+d),n	(IX+d) <-- n
LD (IY+d),n	(IY+d) <-- n
LD A,(BC)	A <-- (BC)
LD A,(DE)	A <-- (DE)
LD A,(nn)	A <-- (nn)
LD (BC),A	(BC) <-- A
LD (DE),A	(DE) <-- A
LD (nn),A	(nn) <-- A
LD A,I	A <-- I
LD A,R	A <-- R
LD I,A	I <-- A
LD R,A	R <-- A
LD dd,nn	dd <-- nn
LD IX,nn	IX <-- nn
LD IY,nn	IY <-- nn

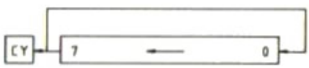
Mnemonic	Symbolische omschrijving	
LD HL,(nn)	H <-- (nn+1) L <-- (nn)	
LD dd,(nn)	ddH <-- (nn+1) ddL <-- (nn)	
LD IX,(nn)	IXH <-- (nn+1) IXL <-- (nn)	
LD IY,(nn)	IYH <-- (nn+1) IYL <-- (nn)	
LD (nn),HL	(nn+1) <-- H (nn) <-- L	
LD (nn),dd	(nn+1) <-- ddH (nn) <-- ddL	
LD (nn),IX	(nn+1) <-- IXH (nn) <-- IXL	
LD (nn),IY	(nn+1) <-- IYH (nn) <-- IYL	
LD SP,HL	SP <-- HL	
LD SP,IX	SP <-- IX	
LD SP,IY	SP <-- IY	
LDD	(DE) <-- (HL) DE <-- DE-1 HL <-- HL-1 BC <-- BC-1	
LDDR	Als LDD, totdat BC=0	
LDI	(DE) <-- (HL) DE <-- DE+1 HL <-- HL+1 BC <-- B-1	
LDIR	Als LDI, totdat BC=0	
NEG	A <-- 0-A	
NOP	Geen bewerking	
OR r	A <-- A ∨ r	
OR n	A <-- A ∨ n	
OR (HL)	A <-- A ∨ (HL)	
OR (IX+d)	A <-- A ∨ (IX+d)	
OR (IY+d)	A <-- A ∨ (IY+d)	
OTDR	(C) <-- (HL) B <-- B-1 HL <-- HL-1 Wordt herhaald totdat B=0	(C naar A0-A7 B naar A8-A15)
OTIR	(C) <-- (HL) B <-- B-1 HL <-- HL+1 Wordt herhaald totdat B=0	(C naar A0-A7 B naar A8-A15)

Mnemonic	Symbolische omschrijving	
OUT (n),A	(n) <-- A	(n naar A0-A7 A naar A8-A15)
OUT (C),r	(C) <-- r	(C naar A0-A7 B naar A8-A15)
OUTD	Als OTDR, doch slechts eenmaal.	
OUTI	Als OTIR, doch slechts eenmaal.	
POP qq	qqH <-- (SP+1) qqL <-- (SP)	
POP IX	IXH <-- (SP+1) IXL <-- (SP)	
POP IY	IYH <-- (SP+1) IYL <-- (SP)	
PUSH qq	(SP-2) <-- qqL (SP-1) <-- qqH	
PUSH IX	(SP-2) <-- IXL (SP-1) <-- IXH	
PUSH IY	(SP-2) <-- IYL (SP-1) <-- IYH	
RES b,r	rb <-- 0	
RES b,(HL)	(HL)b <-- 0	
RES b,(IX+d)	(IX+d)b <-- 0	
RES b,(IY+d)	(IY+d)b <-- 0	
RET	PCL <-- (SP) PCH <-- (SP+1)	
RET cc	Indien cc=waar, dan als RET	
RETI	Return from interrupt	
RETN	Return from non maskable interrupt	

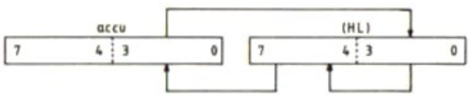
RL/RLA



RLC/RLCA



RLD

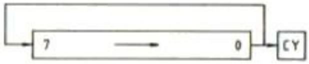


RR/RRA

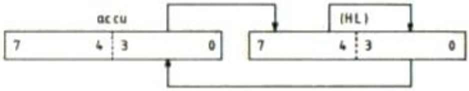


Mnemonic	Symbolische omschrijving

RRC/RRCA



RRD

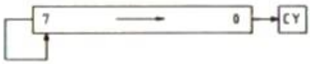


Mnemonic	Symbolische omschrijving
RST p	(SP-1) <-- PCH
	(SP-2) <-- PCL
	PCH <-- 0
	PCL <-- p
SBC r	A <-- A-r-CY
SBC n	A <-- A-n-CY
SBC (HL)	A <-- A-(HL)-CY
SBC (IX+d)	A <-- A-(IX+d)-CY
SBC (IY+d)	A <-- A-(IY+d)-CY
SBC HL,ss	HL <-- HL-ss-CY
SCF	CY <-- 1
SET b,r	rb <-- 1
SET b,(HL)	(HL)b <-- 1
SET b,(IX+d)	(IX+d) <-- 1
SET b,(IY+d)	(IY+d) <-- 1

SLA



SRA



SRL



Mnemonic	Symbolische omschrijving
SUB r	A <-- A-r
SUB n	A <-- A-n
SUB (HL)	A <-- A-(HL)
SUB (IX+d)	A <-- A-(IX+d)
SUB (IY+d)	A <-- A-(IY+d)
XOR r	A <-- A ⊕ r
XOR n	A <-- A ⊕ n
XOR (HL)	A <-- A ⊕ (HL)
XOR (IX+d)	A <-- A ⊕ (IX+d)
XOR (IY+d)	A <-- A ⊕ (IY+d)

cc	Conditie	P	dd	qq
NZ	- non zero	00 hex	BC	BC
Z	- zero	08 hex	DE	DE
NC	- no carry	10 hex	HL	HL
C	- carry	18 hex	IX	AF
PO	- pariteit oneven	20 hex	IY	IX
PE	- pariteit even	28 hex	SP	IY
P	- positief	30 hex		
M	- negatief	38 hex		

ss	r	rr	pp	b
BC	B	BC	BC	0
DE	C	DE	DE	1
HL	D	IY	IX	2
SP	E	SP	SP	3
	H			4
	L			5
	A			6
				7

22 Z80-instructieset op volgorde van mnemonics

In de hiernavolgende tabel zijn in de hexadecimale code van de instructies de volgende voorbeeldwaarden opgenomen:

```
nn - 0584H (dit is in de instructie 8405)
d  - 5     (dit is in de instructie 05)
n  - 20H   (dit is in de instructie 20)
e  - 2EH   (dit is in de instructie 2E)
```

mnemonic	hex.code	mnemonic	hex.code
ADC A,(HL)	8E	ADD IY,DE	FD19
ADC A,(IX+d)	DD8E05	ADD IY,IY	FD29
ADC A,(IY+d)	FD8E05	ADD IY,SP	FD39
ADC A,A	8F	AND (HL)	A6
ADC A,B	88	AND (IX+d)	DDA605
ADC A,C	89	AND (IY+d)	FDA605
ADC A,D	8A	AND A	A7
ADC A,E	8B	AND B	A0
ADC A,H	8C	AND C	A1
ADC A,L	8D	AND D	A2
ADC A,n	CE20	AND E	A3
ADC HL,BC	ED4A	AND H	A4
ADC HL,DE	ED5A	AND L	A5
ADC HL,HL	ED6A	AND n	E620
ADC HL,SP	ED7A	BIT 0,(HL)	CB46
ADD A,(HL)	86	BIT 0,(IX+d)	DDCB0546
ADD A,(IX+d)	DD8605	BIT 0,(IY+d)	FDCB0546
ADD A,(IY+d)	FD8605	BIT 0,A	CB47
ADD A,A	87	BIT 0,B	CB40
ADD A,B	80	BIT 0,C	CB41
ADD A,C	81	BIT 0,D	CB42
ADD A,D	82	BIT 0,E	CB43
ADD A,E	83	BIT 0,H	CB44
ADD A,H	84	BIT 0,L	CB45
ADD A,L	85	BIT 1,(HL)	CB4E
ADD A,n	C620	BIT 1,(IX+d)	DDCB054E
ADD HL,BC	09	BIT 1,(IY+d)	FDCB054E
ADD HL,DE	19	BIT 1,A	CB4F
ADD HL,HL	29	BIT 1,B	CB48
ADD HL,SP	39	BIT 1,C	CB49
ADD IX,BC	DD09	BIT 1,D	CB4A
ADD IX,DE	DD19	BIT 1,E	CB4B
ADD IX,IX	DD29	BIT 1,H	CB4C
ADD IX,SP	DD39	BIT 1,L	CB4D
ADD IY,BC	FD09	BIT 2,(HL)	CB56

mnemonic	hex.code	mnemonic	hex.code
BIT 2,(IX+d)	DDCB0556	BIT 6,C	CB71
BIT 2,(IY+d)	FDCB0556	BIT 6,D	CB72
BIT 2,A	CB57	BIT 6,E	CB73
BIT 2,B	CB50	BIT 6,H	CB74
BIT 2,C	CB51	BIT 6,L	CB75
BIT 2,D	CB52	BIT 7,(HL)	CB7E
BIT 2,E	CB53	BIT 7,(IX+d)	DDCB057E
BIT 2,H	CB54	BIT 7,(IY+d)	FDCB057E
BIT 2,L	CB55	BIT 7,A	CB7F
BIT 3,(HL)	CB5E	BIT 7,B	CB78
BIT 3,(IX+d)	DDCB055E	BIT 7,C	CB79
BIT 3,(IY+d)	FDCB055E	BIT 7,D	CB7A
BIT 3,A	CB5F	BIT 7,E	CB7B
BIT 3,B	CB58	BIT 7,H	CB7C
BIT 3,C	CB59	BIT 7,L	CB7D
BIT 3,D	CB5A	CALL C,nn	DC8405
BIT 3,E	CB5B	CALL M,nn	FC8405
BIT 3,H	CB5C	CALL NC,nn	D48405
BIT 3,L	CB5D	CALL NZ,nn	C48405
BIT 4,(HL)	CB66	CALL P,nn	F48405
BIT 4,(IX+d)	DDCB0566	CALL PE,nn	EC8405
BIT 4,(IY+d)	FDCB0566	CALL PO,nn	E48405
BIT 4,A	CB67	CALL Z,nn	CC8405
BIT 4,B	CB60	CALL nn	CD8405
BIT 4,C	CB61	CCF	3F
BIT 4,D	CB62	CP (HL)	BE
BIT 4,E	CB63	CP (IX+d)	DDBE05
BIT 4,H	CB64	CP (IY+d)	FDBE05
BIT 4,L	CB65	CP A	BF
BIT 5,(HL)	CB6E	CP B	B8
BIT 5,(IX+d)	DDCB056E	CP C	B9
BIT 5,(IY+d)	FDCB056E	CP D	BA
BIT 5,A	CB6F	CP E	BB
BIT 5,B	CB68	CP H	BC
BIT 5,C	CB69	CP L	BD
BIT 5,D	CB6A	CP n	FE20
BIT 5,E	CB6B	CPD	EDA9
BIT 5,H	CB6C	CPDR	ED89
BIT 5,L	CB6D	CPI	EDA1
BIT 6,(HL)	CB76	CPIR	ED81
BIT 6,(IX+d)	DDCB0576	CPL	2F
BIT 6,(IY+d)	FDCB0576	DAA	27
BIT 6,A	CB77	DEC (HL)	35
BIT 6,B	CB70	DEC (IX+d)	DD3505

mnemonic		hex.code	mnemonic		hex.code
DEC	(IY+d)	FD3505	INC	H	24
DEC	A	3D	INC	HL	23
DEC	B	05	INC	IX	DD23
DEC	BC	0B	INC	IY	FD23
DEC	C	0D	INC	L	2C
DEC	D	15	INC	SP	33
DEC	DE	1B	IN	A,(n)	DB20
DEC	E	1D	IND		EDAA
DEC	H	25	INDR		EDBA
DEC	HL	2B	INI		EDA2
DEC	IX	DD2B	INIR		EDB2
DEC	IY	FD2B	JP	nn	C38405
DEC	L	2D	JP	(HL)	E9
DEC	SP	3B	JP	(IX)	DDE9
DI		F3	JP	(IY)	FDE9
DJNZ	e	102E	JP	C,nn	DA8405
EI		FB	JP	M,nn	FA8405
EX	(SP),HL	E3	JP	NC,nn	D28405
EX	(SP),IX	DDE3	JP	NZ,nn	C28405
EX	(SP),IY	FDE3	JP	P,nn	F28405
EX	AF,AF'	08	JP	PE,nn	EA8405
EX	DE,HL	EB	JP	PO,nn	E28405
EXX		D9	JP	Z,nn	CA8405
HALT		76	JR	C,e	382E
IM	0	ED46	JR	NC,e	302E
IM	1	ED56	JR	NZ,e	202E
IM	2	ED5E	JR	Z,e	282E
IN	A,(C)	ED78	JR	e	182E
IN	B,(C)	ED40	LD	(BC),A	02
IN	C,(C)	ED48	LD	(DE),A	12
IN	D,(C)	ED50	LD	(HL),A	77
IN	E,(C)	ED58	LD	(HL),B	70
IN	H,(C)	ED60	LD	(HL),C	71
IN	L,(C)	ED68	LD	(HL),D	72
INC	(HL)	34	LD	(HL),E	73
INC	(IX+d)	DD3405	LD	(HL),H	74
INC	(IY+d)	FD3405	LD	(HL),L	75
INC	A	3C	LD	(HL),n	3620
INC	B	04	LD	(IX+d),A	DD7705
INC	BC	03	LD	(IX+d),B	DD7005
INC	C	0C	LD	(IX+d),C	DD7105
INC	D	14	LD	(IX+d),D	DD7205
INC	DE	13	LD	(IX+d),E	DD7305
INC	E	1C	LD	(IX+d),H	DD7405

mnemonic		hex.code	mnemonic		hex.code
LD	(IX+d),L	DD7505	LD	BC,(nn)	ED4B8405
LD	(IX+d),n	DD360520	LD	BC,nn	018405
LD	(IY+d),A	FD7705	LD	C,(HL)	4E
LD	(IY+d),B	FD7005	LD	C,(IX+d)	DD4E05
LD	(IY+d),C	FD7105	LD	C,(IY+d)	FD4E05
LD	(IY+d),D	FD7205	LD	C,A	4F
LD	(IY+d),E	FD7305	LD	C,B	48
LD	(IY+d),H	FD7405	LD	C,C	49
LD	(IY+d),L	FD7505	LD	C,D	4A
LD	(IY+d),n	FD360520	LD	C,E	4B
LD	(nn),A	328405	LD	C,H	4C
LD	(nn),BC	ED438405	LD	C,L	4D
LD	(nn),DE	ED538405	LD	C,n	0E20
LD	(nn),HL	228405	LD	D,(HL)	56
LD	(nn),IX	DD228405	LD	D,(IX+d)	DD5605
LD	(nn),IY	FD228405	LD	D,(IY+d)	FD5605
LD	(nn),SP	ED738405	LD	D,A	57
LD	A,(BC)	0A	LD	D,B	50
LD	A,(DE)	1A	LD	D,C	51
LD	A,(HL)	7E	LD	D,D	52
LD	A,(IX+d)	DD7E05	LD	D,E	53
LD	A,(IY+d)	FD7E05	LD	D,H	54
LD	A,(nn)	3A8405	LD	D,L	55
LD	A,A	7F	LD	D,n	1620
LD	A,B	78	LD	DE,(nn)	ED5B8405
LD	A,C	79	LD	DE,nn	118405
LD	A,D	7A	LD	E,(HL)	5E
LD	A,E	7B	LD	E,(IX+d)	DD5E05
LD	A,H	7C	LD	E,(IY+d)	FD5E05
LD	A,I	ED57	LD	E,A	5F
LD	A,L	7D	LD	E,B	58
LD	A,n	3E20	LD	E,C	59
LD	A,R	ED5F	LD	E,D	5A
LD	B,(HL)	46	LD	E,E	5B
LD	B,(IX+d)	DD4605	LD	E,H	5C
LD	B,(IY+d)	FD4605	LD	E,L	5D
LD	B,A	47	LD	E,n	1E20
LD	B,B	40	LD	H,(HL)	66
LD	B,C	41	LD	H,(IX+d)	DD6605
LD	B,D	42	LD	H,(IY+d)	FD6605
LD	B,E	43	LD	H,A	67
LD	B,H	44	LD	H,B	60
LD	B,L	45	LD	H,C	61
LD	B,n	0620	LD	H,D	62

mnemonic		hex.code	mnemonic		hex.code
LD	H,E	63	OR	n	F620
LD	H,H	64	OTDR		ED8B
LD	H,L	65	OTIR		EDB3
LD	H,n	2620	OUT	(C),A	ED79
LD	HL,(nn)	2A8405	OUT	(C),B	ED41
LD	HL,nn	218405	OUT	(C),C	ED49
LD	I,A	ED47	OUT	(C),D	ED51
LD	IX,(nn)	DD2A8405	OUT	(C),E	ED59
LD	IX,nn	DD218405	OUT	(C),H	ED61
LD	IY,(nn)	FD2A8405	OUT	(C),L	ED69
LD	IY,nn	FD218405	OUT	(n),A	D320
LD	L,(HL)	6E	OUTD		EDAB
LD	L,(IX+d)	DD6E05	OUTI		EDA3
LD	L,(IY+d)	FD6E05	POP	AF	F1
LD	L,A	6F	POP	BC	C1
LD	L,B	68	POP	DE	D1
LD	L,C	69	POP	HL	E1
LD	L,D	6A	POP	IX	DDE1
LD	L,E	6B	POP	IY	FDE1
LD	L,H	6C	PUSH	AF	F5
LD	L,L	6D	PUSH	BC	C5
LD	L,n	2E20	PUSH	DE	D5
LD	R,A	ED4F	PUSH	HL	E5
LD	SP,(nn)	ED7B8405	PUSH	IX	DDE5
LD	SP,HL	F9	PUSH	IY	FDE5
LD	SP,IX	DDF9	RES	0,(HL)	CB86
LD	SP,IY	FDF9	RES	0,(IX+d)	DDCB0586
LD	SP,nn	318405	RES	0,(IY+d)	FDCB0586
LDD		EDA8	RES	0,A	CB87
LDDR		EDB8	RES	0,B	CB80
LDI		EDA0	RES	0,C	CB81
LDIR		EDB0	RES	0,D	CB82
NEG		ED44	RES	0,E	CB83
NOP		00	RES	0,H	CB84
OR	(HL)	B6	RES	0,L	CB85
OR	(IX+d)	DDB605	RES	1,(HL)	CB8E
OR	(IY+d)	FDB605	RES	1,(IX+d)	DDCB058E
OR	A	B7	RES	1,(IY+d)	FDCB058E
OR	B	B0	RES	1,A	CB8F
OR	C	B1	RES	1,B	CB88
OR	D	B2	RES	1,C	CB89
OR	E	B3	RES	1,D	CB8A
OR	H	B4	RES	1,E	CB8B
OR	L	B5	RES	1,H	CB8C

mnemonic		hex.code	mnemonic		hex.code
RES	1,L	CB8D	RES	6,A	CBB7
RES	2,(HL)	CB96	RES	6,B	CBB0
RES	2,(IX+d)	DDCB0596	RES	6,C	CBB1
RES	2,(IY+d)	FDCB0596	RES	6,D	CBB2
RES	2,A	CB97	RES	6,E	CBB3
RES	2,B	CB90	RES	6,H	CBB4
RES	2,C	CB91	RES	6,L	CBB5
RES	2,D	CB92	RES	7,(HL)	CBBE
RES	2,E	CB93	RES	7,(IX+d)	DDCB05BE
RES	2,H	CB94	RES	7,(IY+d)	FDCB05BE
RES	2,L	CB95	RES	7,A	CBBF
RES	3,(HL)	CB9E	RES	7,B	CBB8
RES	3,(IX+d)	DDCB059E	RES	7,C	CBB9
RES	3,(IY+d)	FDCB059E	RES	7,D	CBBA
RES	3,A	CB9F	RES	7,E	CBBB
RES	3,B	CB98	RES	7,H	CBBC
RES	3,C	CB99	RES	7,L	CBBD
RES	3,D	CB9A	RET		C9
RES	3,E	CB9B	RET	C	D8
RES	3,H	CB9C	RET	M	F8
RES	3,L	CB9D	RET	NC	D0
RES	4,(HL)	CBA6	RET	NZ	C0
RES	4,(IX+d)	DDCB05A6	RET	P	F0
RES	4,(IY+d)	FDCB05A6	RET	PE	E8
RES	4,A	CBA7	RET	PO	E0
RES	4,B	CBA0	RET	Z	C8
RES	4,C	CBA1	RETI		ED4D
RES	4,D	CBA2	RETN		ED45
RES	4,E	CBA3	RL	(HL)	CB16
RES	4,H	CBA4	RL	(IX+d)	DDCB0516
RES	4,L	CBA5	RL	(IY+d)	FDCB0516
RES	5,(HL)	CBAE	RL	A	CB17
RES	5,(IX+d)	DDCB05AE	RL	B	CB10
RES	5,(IY+d)	FDCB05AE	RL	C	CB11
RES	5,A	CBAF	RL	D	CB12
RES	5,B	CBA8	RL	E	CB13
RES	5,C	CBA9	RL	H	CB14
RES	5,D	CBAA	RL	L	CB15
RES	5,E	CBAB	RLA		17
RES	5,H	CBAC	RLC	(HL)	CB06
RES	5,L	CBAD	RLC	(IX+d)	DDCB0506
RES	6,(HL)	CBB6	RLC	(IY+d)	FDCB0506
RES	6,(IX+d)	DDCB05B6	RLC	A	CB07
RES	6,(IY+d)	FDCB05B6	RLC	B	CB00

mnemonic	hex.code	mnemonic	hex.code
RLC C	CB01	SBC A,C	99
RLC D	CB02	SBC A,D	9A
RLC E	CB03	SBC A,E	9B
RLC H	CB04	SBC A,H	9C
RLC L	CB05	SBC A,L	9D
RLCA	07	SBC HL,BC	ED42
RLD	ED6F	SBC HL,DE	ED52
RR (HL)	CB1E	SBC HL,HL	ED62
RR (IX+d)	DDCB051E	SBC HL,SP	ED72
RR (IY+d)	FDCB051E	SCF	37
RR A	CB1F	SET 0,(HL)	CBC6
RR B	CB18	SET 0,(IX+d)	DDCB05C6
RR C	CB19	SET 0,(IY+d)	FDCB05C6
RR D	CB1A	SET 0,A	CBC7
RR E	CB1B	SET 0,B	CBC0
RR H	CB1C	SET 0,C	CBC1
RR L	CB1D	SET 0,D	CBC2
RRA	1F	SET 0,E	CBC3
RRC (HL)	CB0E	SET 0,H	CBC4
RRC (IX+d)	DDCB050E	SET 0,L	CBC5
RRC (IY+d)	FDCB050E	SET 1,(HL)	CBCE
RRC A	CB0F	SET 1,(IX+d)	DDCB05CE
RRC B	CB08	SET 1,(IY+d)	FDCB05CE
RRC C	CB09	SET 1,A	CBCF
RRC D	CB0A	SET 1,B	CBC8
RRC E	CB0B	SET 1,C	CBC9
RRC H	CB0C	SET 1,D	CBCA
RRC L	CB0D	SET 1,E	CBCB
RRCA	0F	SET 1,H	CBCC
RRD	ED67	SET 1,L	CBCD
RST 00H	C7	SET 2,(HL)	CBD6
RST 08H	CF	SET 2,(IX+d)	DDCB05D6
RST 10H	D7	SET 2,(IY+d)	FDCB05D6
RST 18H	DF	SET 2,A	CBD7
RST 20H	E7	SET 2,B	CBD0
RST 28H	EF	SET 2,C	CBD1
RST 30H	F7	SET 2,D	CBD2
RST 38H	FF	SET 2,E	CBD3
SBC A,n	DE20	SET 2,H	CBD4
SBC A,(HL)	9E	SET 2,L	CBD5
SBC A,(IX+d)	DD9E05	SET 3,(HL)	CBDE
SBC A,(IY+d)	FD9E05	SET 3,(IX+d)	DDCB05DE
SBC A,A	9F	SET 3,(IY+d)	FDCB05DE
SBC A,B	98	SET 3,A	CBDF

mnemonic	hex.code	mnemonic	hex.code
SET 3,B	CBD8	SET 7,H	CBFC
SET 3,C	CBD9	SET 7,L	CBFD
SET 3,D	CBDA	SLA (HL)	CB26
SET 3,E	CBDB	SLA (IX+d)	DDCB0526
SET 3,H	CBDC	SLA (IY+d)	FDCB0526
SET 3,L	CBDD	SLA A	CB27
SET 4,(HL)	CBE6	SLA B	CB20
SET 4,(IX+d)	DDCB05E6	SLA C	CB21
SET 4,(IY+d)	FDCB05E6	SLA D	CB22
SET 4,A	CBE7	SLA E	CB23
SET 4,B	CBE0	SLA H	CB24
SET 4,C	CBE1	SLA L	CB25
SET 4,D	CBE2	SRA (HL)	CB2E
SET 4,E	CBE3	SRA (IX+d)	DDCB052E
SET 4,H	CBE4	SRA (IY+d)	FDCB052E
SET 4,L	CBE5	SRA A	CB2F
SET 5,(HL)	CBEE	SRA B	CB28
SET 5,(IX+d)	DDCB05EE	SRA C	CB29
SET 5,(IY+d)	FDCB05EE	SRA D	CB2A
SET 5,A	CBEF	SRA E	CB2B
SET 5,B	CBE8	SRA H	CB2C
SET 5,C	CBE9	SRA L	CB2D
SET 5,D	CBEA	SRL (HL)	CB3E
SET 5,E	CBEB	SRL (IX+d)	DDCB053E
SET 5,H	CBEC	SRL (IY+d)	FDCB053E
SET 5,L	CBED	SRL A	CB3F
SET 6,(HL)	CBF6	SRL B	CB38
SET 6,(IX+d)	DDCB05F6	SRL C	CB39
SET 6,(IY+d)	FDCB05F6	SRL D	CB3A
SET 6,A	CBF7	SRL E	CB3B
SET 6,B	CBF0	SRL H	CB3C
SET 6,C	CBF1	SRL L	CB3D
SET 6,D	CBF2	SUB (HL)	96
SET 6,E	CBF3	SUB (IX+d)	DD9605
SET 6,H	CBF4	SUB (IY+d)	FD9605
SET 6,L	CBF5	SUB A	97
SET 7,(HL)	CBFE	SUB B	90
SET 7,(IX+d)	DDCB05FE	SUB C	91
SET 7,(IY+d)	FDCB05FE	SUB D	92
SET 7,A	CBFF	SUB E	93
SET 7,B	CBF8	SUB H	94
SET 7,C	CBF9	SUB L	95
SET 7,D	CBFA	SUB n	D620
SET 7,E	CBFB	XOR (HL)	AE

mnemonic	hex.code
XOR (IX+d)	DDAE05
XOR (IY+d)	FDAE05
XOR A	AF
XOR B	A8
XOR C	A9
XOR D	AA
XOR E	AB
XOR H	AC
XOR L	AD
XOR n	EE20

23 Z80-instructieset op volgorde van hexcode

In de hierna volgende tabel zijn in de hexadecimale codes van de instructies de volgende voorbeeldwaarden opgenomen:

```
nn - 0584H (dit is in de instructie 8405)
d  - 5     (dit is in de instructie 05)
n  - 20H   (dit is in de instructie 20)
e  - 2EH   (dit is in de instructie 2E)
```

hex.code	mnemonic		hex.code	mnemonic	
00	NOP		23	INC	HL
018405	LD	BC,nn	24	INC	H
02	LD	(BC),A	25	DEC	H
03	INC	BC	2620	LD	H,n
04	INC	B	27	DAA	
05	DEC	B	282E	JR	Z,e
0620	LD	B,n	29	ADD	HL,HL
07	RLCA		2A8405	LD	HL,(nn)
08	EX	AF,AF'	2B	DEC	HL
09	ADD	HL,BC	2C	INC	L
0A	LD	A,(BC)	2D	DEC	L
0B	DEC	BC	2E20	LD	L,n
0C	INC	C	2F	CPL	
0D	DEC	C	302E	JR	NC,e
0E20	LD	C,n	318405	LD	SP,nn
0F	RRCA		328405	LD	(nn),A
102E	DJNZ	e	33	INC	SP
118405	LD	DE,nn	34	INC	(HL)
12	LD	(DE),A	35	DEC	(HL)
13	INC	DE	3620	LD	(HL),n
14	INC	D	37	SCF	
15	DEC	D	382E	JR	C,e
1620	LD	D,n	39	ADD	HL,SP
17	RLA		3A8405	LD	A,(nn)
182E	JR	e	3B	DEC	SP
19	ADD	HL,DE	3C	INC	A
1A	LD	A,(DE)	3D	DEC	A
1B	DEC	DE	3E20	LD	A,n
1C	INC	E	3F	CCF	
1D	DEC	E	40	LD	B,B
1E20	LD	E,n	41	LD	B,C
1F	RRA		42	LD	B,D
202E	JR	NZ,e	43	LD	B,E
218405	LD	HL,nn	44	LD	B,H
228405	LD	(nn),HL	45	LD	B,L

hex.code	mnemonic		hex.code	mnemonic	
46	LD	B,(HL)	72	LD	(HL),D
47	LD	B,A	73	LD	(HL),E
48	LD	C,B	74	LD	(HL),H
49	LD	C,C	75	LD	(HL),L
4A	LD	C,D	76	HALT	
4B	LD	C,E	77	LD	(HL),A
4C	LD	C,H	78	LD	A,B
4D	LD	C,L	79	LD	A,C
4E	LD	C,(HL)	7A	LD	A,D
4F	LD	C,A	7B	LD	A,E
50	LD	D,B	7C	LD	A,H
51	LD	D,C	7D	LD	A,L
52	LD	D,D	7E	LD	A,(HL)
53	LD	D,E	7F	LD	A,A
54	LD	D,H	80	ADD	A,B
55	LD	D,L	81	ADD	A,C
56	LD	D,(HL)	82	ADD	A,D
57	LD	D,A	83	ADD	A,E
58	LD	E,B	84	ADD	A,H
59	LD	E,C	85	ADD	A,L
5A	LD	E,D	86	ADD	A,(HL)
5B	LD	E,E	87	ADD	A,A
5C	LD	E,H	88	ADC	A,B
5D	LD	E,L	89	ADC	A,C
5E	LD	E,(HL)	8A	ADC	A,D
5F	LD	E,A	8B	ADC	A,E
60	LD	H,B	8C	ADC	A,H
61	LD	H,C	8D	ADC	A,L
62	LD	H,D	8E	ADC	A,(HL)
63	LD	H,E	8F	ADC	A,A
64	LD	H,H	90	SUB	B
65	LD	H,L	91	SUB	C
66	LD	H,(HL)	92	SUB	D
67	LD	H,A	93	SUB	E
68	LD	L,B	94	SUB	H
69	LD	L,C	95	SUB	L
6A	LD	L,D	96	SUB	(HL)
6B	LD	L,E	97	SUB	A
6C	LD	L,H	98	SBC	A,B
6D	LD	L,L	99	SBC	A,C
6E	LD	L,(HL)	9A	SBC	A,D
6F	LD	L,A	9B	SBC	A,E
70	LD	(HL),B	9C	SBC	A,H
71	LD	(HL),C	9D	SBC	A,L

hex.code	mnemonic		hex.code	mnemonic	
9E	SBC	A,(HL)	CA8405	JP	Z,nn
9F	SBC	A,A	CB00	RLC	B
A0	AND	B	CB01	RLC	C
A1	AND	C	CB02	RLC	D
A2	AND	D	CB03	RLC	E
A3	AND	E	CB04	RLC	H
A4	AND	H	CB05	RLC	L
A5	AND	L	CB06	RLC	(HL)
A6	AND	(HL)	CB07	RLC	A
A7	AND	A	CB08	RRC	B
A8	XOR	B	CB09	RRC	C
A9	XOR	C	CB0A	RRC	D
AA	XOR	D	CB0B	RRC	E
AB	XOR	E	CB0C	RRC	H
AC	XOR	H	CB0D	RRC	L
AD	XOR	L	CB0E	RRC	(HL)
AE	XOR	(HL)	CB0F	RRC	A
AF	XOR	A	CB10	RL	B
B0	OR	B	CB11	RL	C
B1	OR	C	CB12	RL	D
B2	OR	D	CB13	RL	E
B3	OR	E	CB14	RL	H
B4	OR	H	CB15	RL	L
B5	OR	L	CB16	RL	(HL)
B6	OR	(HL)	CB17	RL	A
B7	OR	A	CB18	RR	B
B8	CP	B	CB19	RR	C
B9	CP	C	CB1A	RR	D
BA	CP	D	CB1B	RR	E
BB	CP	E	CB1C	RR	H
BC	CP	H	CB1D	RR	L
BD	CP	L	CB1E	RR	(HL)
BE	CP	(HL)	CB1F	RR	A
BF	CP	A	CB20	SLA	B
C0	RET	NZ	CB21	SLA	C
C1	POP	BC	CB22	SLA	D
C28405	JP	NZ,nn	CB23	SLA	E
C38405	JP	nn	CB24	SLA	H
C48405	CALL	NZ,nn	CB25	SLA	L
C5	PUSH	BC	CB26	SLA	(HL)
C620	ADD	A,n	CB27	SLA	A
C7	RST	0	CB28	SRA	B
C8	RET	Z	CB29	SRA	C
C9	RET		CB2A	SRA	D

hex.code	mnemonic	hex.code	mnemonic
CB2B	SRA E	CB5F	BIT 3,A
CB2C	SRA H	CB60	BIT 4,B
CB2D	SRA L	CB61	BIT 4,C
CB2E	SRA (HL)	CB62	BIT 4,D
CB2F	SRA A	CB63	BIT 4,E
CB38	SRL B	CB64	BIT 4,H
CB39	SRL C	CB65	BIT 4,L
CB3A	SRL D	CB66	BIT 4,(HL)
CB3B	SRL E	CB67	BIT 4,A
CB3C	SRL H	CB68	BIT 5,B
CB3D	SRL L	CB69	BIT 5,C
CB3E	SRL (HL)	CB6A	BIT 5,D
CB3F	SRL A	CB6B	BIT 5,E
CB40	BIT 0,B	CB6C	BIT 5,H
CB41	BIT 0,C	CB6D	BIT 5,L
CB42	BIT 0,D	CB6E	BIT 5,(HL)
CB43	BIT 0,E	CB6F	BIT 5,A
CB44	BIT 0,H	CB70	BIT 6,B
CB45	BIT 0,L	CB71	BIT 6,C
CB46	BIT 0,(HL)	CB72	BIT 6,D
CB47	BIT 0,A	CB73	BIT 6,E
CB48	BIT 1,B	CB74	BIT 6,H
CB49	BIT 1,C	CB75	BIT 6,L
CB4A	BIT 1,D	CB76	BIT 6,(HL)
CB4B	BIT 1,E	CB77	BIT 6,A
CB4C	BIT 1,H	CB78	BIT 7,B
CB4D	BIT 1,L	CB79	BIT 7,C
CB4E	BIT 1,(HL)	CB7A	BIT 7,D
CB4F	BIT 1,A	CB7B	BIT 7,E
CB50	BIT 2,B	CB7C	BIT 7,H
CB51	BIT 2,C	CB7D	BIT 7,L
CB52	BIT 2,D	CB7E	BIT 7,(HL)
CB53	BIT 2,E	CB7F	BIT 7,A
CB54	BIT 2,H	CB80	RES 0,B
CB55	BIT 2,L	CB81	RES 0,C
CB56	BIT 2,(HL)	CB82	RES 0,D
CB57	BIT 2,A	CB83	RES 0,E
CB58	BIT 3,B	CB84	RES 0,H
CB59	BIT 3,C	CB85	RES 0,L
CB5A	BIT 3,D	CB86	RES 0,(HL)
CB5B	BIT 3,E	CB87	RES 0,A
CB5C	BIT 3,H	CB88	RES 1,B
CB5D	BIT 3,L	CB89	RES 1,C
CB5E	BIT 3,(HL)	CB8A	RES 1,D

hex.code	mnemonic	hex.code	mnemonic
CB8B	RES 1,E	CBB7	RES 6,A
CB8C	RES 1,H	CBB8	RES 7,B
CB8D	RES 1,L	CBB9	RES 7,C
CB8E	RES 1,(HL)	CBBA	RES 7,D
CB8F	RES 1,A	CBBB	RES 7,E
CB90	RES 2,B	CBBC	RES 7,H
CB91	RES 2,C	CBBD	RES 7,L
CB92	RES 2,D	CBBE	RES 7,(HL)
CB93	RES 2,E	CBBF	RES 7,A
CB94	RES 2,H	CBC0	SET 0,B
CB95	RES 2,L	CBC1	SET 0,C
CB96	RES 2,(HL)	CBC2	SET 0,D
CB97	RES 2,A	CBC3	SET 0,E
CB98	RES 3,B	CBC4	SET 0,H
CB99	RES 3,C	CBC5	SET 0,L
CB9A	RES 3,D	CBC6	SET 0,(HL)
CB9B	RES 3,E	CBC7	SET 0,A
CB9C	RES 3,H	CBC8	SET 1,B
CB9D	RES 3,L	CBC9	SET 1,C
CB9E	RES 3,(HL)	CBCA	SET 1,D
CB9F	RES 3,A	CBCB	SET 1,E
CBA0	RES 4,B	CBCC	SET 1,H
CBA1	RES 4,C	CBCD	SET 1,L
CBA2	RES 4,D	CBCE	SET 1,(HL)
CBA3	RES 4,E	CBCF	SET 1,A
CBA4	RES 4,H	CBD0	SET 2,B
CBA5	RES 4,L	CBD1	SET 2,C
CBA6	RES 4,(HL)	CBD2	SET 2,D
CBA7	RES 4,A	CBD3	SET 2,E
CBA8	RES 5,B	CBD4	SET 2,H
CBA9	RES 5,C	CBD5	SET 2,L
CBAA	RES 5,D	CBD6	SET 2,(HL)
CBAB	RES 5,E	CBD7	SET 2,A
CBAC	RES 5,H	CBD8	SET 3,B
CBAD	RES 5,L	CBD9	SET 3,C
CBAE	RES 5,(HL)	CBDA	SET 3,D
CBAF	RES 5,A	CBDB	SET 3,E
CBB0	RES 6,B	CBDC	SET 3,H
CBB1	RES 6,C	CBDD	SET 3,L
CBB2	RES 6,D	CBDE	SET 3,(HL)
CBB3	RES 6,E	CBDF	SET 3,A
CBB4	RES 6,H	CBE0	SET 4,B
CBB5	RES 6,L	CBE1	SET 4,C
CBB6	RES 6,(HL)	CBE2	SET 4,D

hex.code	mnemonic	
CBE3	SET	4,E
CBE4	SET	4,H
CBE5	SET	4,L
CBE6	SET	4,(HL)
CBE7	SET	4,A
CBE8	SET	5,B
CBE9	SET	5,C
CBEA	SET	5,D
CBEB	SET	5,E
CBEC	SET	5,H
CBED	SET	5,L
CBEE	SET	5,(HL)
CBEF	SET	5,A
CBF0	SET	6,B
CBF1	SET	6,C
CBF2	SET	6,D
CBF3	SET	6,E
CBF4	SET	6,H
CBF5	SET	6,L
CBF6	SET	6,(HL)
CBF7	SET	6,A
CBF8	SET	7,B
CBF9	SET	7,C
CBFA	SET	7,D
CBFB	SET	7,E
CBFC	SET	7,H
CBFD	SET	7,L
CBFE	SET	7,(HL)
CBFF	SET	7,A
CC8405	CALL	Z,nn
CD8405	CALL	nn
CE20	ADC	A,n
CF	RST	8
D0	RET	NC
D1	POP	DE
D28405	JP	NC,nn
D320	OUT	n,A
D48405	CALL	NC,nn
D5	PUSH	DE
D620	SUB	n
D7	RST	10
D8	RET	C
D9	EXX	
DA8405	JP	C,nn
DB20	IN	A,n
DC8405	CALL	C,nn
DD09	ADD	IX,BC
DD19	ADD	IX,DE
DD218405	LD	IX,nn
DD228405	LD	(nn),IX
DD23	INC	IX
DD29	ADD	IX,IX
DD2A8405	LD	IX,(nn)
DD2B	DEC	IX
DD3405	INC	(IX+d)
DD3505	DEC	(IX+d)
DD360520	LD	(IX+d),n
DD39	ADD	IX,SP
DD4605	LD	B,(IX+d)
DD4E05	LD	C,(IX+d)
DD5605	LD	D,(IX+d)
DD5E05	LD	E,(IX+d)
DD6605	LD	H,(IX+d)
DD6E05	LD	L,(IX+d)
DD7005	LD	(IX+d),B
DD7105	LD	(IX+d),C
DD7205	LD	(IX+d),D
DD7305	LD	(IX+d),E
DD7405	LD	(IX+d),H
DD7505	LD	(IX+d),L
DD7705	LD	(IX+d),A
DD7E05	LD	A,(IX+d)
DD8605	ADD	A,(IX+d)
DD8E05	ADC	A,(IX+d)
DD9605	SUB	(IX+d)
DD9E05	SBC	A,(IX+d)
DDA605	AND	(IX+d)
DDAE05	XOR	(IX+d)
DDB605	OR	(IX+d)
DDBE05	CP	(IX+d)
DDCB0506	RLC	(IX+d)
DDCB050E	RLC	(IX+d)
DDCB0516	RL	(IX+d)
DDCB051E	RR	(IX+d)
DDCB0526	SLA	(IX+d)
DDCB052E	SRA	(IX+d)
DDCB053E	SRL	(IX+d)
DDCB0546	BIT	0,(IX+d)

hex.code	mnemonic		hex.code	mnemonic	
DDCB054E	BIT	1,(IX+d)	ED41	OUT	(C),B
DDCB0556	BIT	2,(IX+d)	ED42	SBC	HL,BC
DDCB055E	BIT	3,(IX+d)	ED438405	LD	(nn),BC
DDCB0566	BIT	4,(IX+d)	ED44	NEG	
DDCB056E	BIT	5,(IX+d)	ED45	RETN	
DDCB0576	BIT	6,(IX+d)	ED46	IM	0
DDCB057E	BIT	7,(IX+d)	ED47	LD	I,A
DDCB0586	RES	0,(IX+d)	ED48	IN	C,(C)
DDCB058E	RES	1,(IX+d)	ED49	OUT	(C),C
DDCB0596	RES	2,(IX+d)	ED4A	ADC	HL,BC
DDCB059E	RES	3,(IX+d)	ED4B8405	LD	BC,(nn)
DDCB05A6	RES	4,(IX+d)	ED4D	RETI	
DDCB05AE	RES	5,(IX+d)	ED50	IN	D,(C)
DDCB05B6	RES	6,(IX+d)	ED51	OUT	(C),D
DDCB05BE	RES	7,(IX+d)	ED52	SBC	HL,DE
DDCB05C6	SET	0,(IX+d)	ED538405	LD	(nn),DE
DDCB05CE	SET	1,(IX+d)	ED56	IM	1
DDCB05D6	SET	2,(IX+d)	ED57	LD	A,I
DDCB05DE	SET	3,(IX+d)	ED58	IN	E,(C)
DDCB05E6	SET	4,(IX+d)	ED59	OUT	(C),E
DDCB05EE	SET	5,(IX+d)	ED5A	ADC	HL,DE
DDCB05F6	SET	6,(IX+d)	ED5B8405	LD	DE,(nn)
DDCB05FE	SET	7,(IX+d)	ED5E	IM	2
DDE1	POP	IX	ED60	IN	H,(C)
DDE3	EX	(SP),IX	ED61	OUT	(C),H
DDE5	PUSH	IX	ED62	SBC	HL,HL
DDE9	JP	(IX)	ED67	RRD	
DDF9	LD	SP,IX	ED68	IN	L,(C)
DD20	SBC	A,n	ED69	OUT	(C),L
DF	RST	18	ED6A	ADC	HL,HL
E0	RET	PO	ED6F	RLD	
E1	POP	HL	ED72	SBC	HL,SP
E28405	JP	PO,nn	ED738405	LD	(nn),SP
E3	EX	(SP),HL	ED78	IN	A,(C)
E48405	CALL	PO,nn	ED79	OUT	(C),A
E5	PUSH	HL	ED7A	ADC	HL,SP
E620	AND	n	ED7B8405	LD	SP,(nn)
E7	RST	20	EDA0	LDI	
E8	RET	PE	EDA1	CPI	
E9	JP	(HL)	EDA2	INI	
EA8405	JP	PE,nn	EDA3	OUTI	
EB	EX	DE,nn	EDA8	LDD	
EC8405	CALL	PE,nn	EDA9	CPD	
ED40	IN	B,(C)	EDAA	IND	

hex.code	mnemonic		hex.code	mnemonic	
EDAB	OUTD		FD7205	LD	(IY+d),D
EDB0	LDIR		FD7305	LD	(IY+d),E
EDB1	CPIR		FD7405	LD	(IY+d),H
EDB2	INIR		FD7505	LD	(IY+d),L
EDB3	OTIR		FD7705	LD	(IY+d),A
EDB8	LDDR		FD7E05	LD	A,(IY+d)
EDB9	CPDR		FD8605	ADD	A,(IY+d)
EDBA	INDR		FD8E05	ADC	A,(IY+d)
EDBB	OTDR		FD9605	SUB	(IY+d)
EE20	XOR	n	FD9E05	SBC	A,(IY+d)
EF	RST	28	FDA605	AND	(IY+d)
F0	RET	P	FDAE05	XOR	(IY+d)
F1	POP	AF	FDB605	OR	(IY+d)
F28405	JP	P,nn	FDBE05	CP	(IY+d)
F3	DI		FDCB0506	RLC	(IY+d)
F48405	CALL	P,nn	FDCB050E	RRC	(IY+d)
F5	PUSH	AF	FDCB0516	RL	(IY+d)
F620	OR	n	FDCB051E	RR	(IY+d)
F7	RST	30	FDCB0526	SLA	(IY+d)
F8	RET	M	FDCB052E	SRA	(IY+d)
F9	LD	SP,HL	FDCB053E	SRL	(IY+d)
FA8405	JP	M,nn	FDCB0546	BIT	0,(IY+d)
FB	EI		FDCB054E	BIT	1,(IY+d)
FC8405	CALL	M,nn	FDCB0556	BIT	2,(IY+d)
FD09	ADD	IY,BC	FDCB055E	BIT	3,(IY+d)
FD19	ADD	IY,DE	FDCB0566	BIT	4,(IY+d)
FD218405	LD	IY,nn	FDCB056E	BIT	5,(IY+d)
FD228405	LD	(nn),IY	FDCB0576	BIT	6,(IY+d)
FD23	INC	IY	FDCB057E	BIT	7,(IY+d)
FD29	ADD	IY,IY	FDCB0586	RES	0,(IY+d)
FD2A8405	LD	IY,(nn)	FDCB058E	RES	1,(IY+d)
FD2B	DEC	IY	FDCB0596	RES	2,(IY+d)
FD3405	INC	(IY+d)	FDCB059E	RES	3,(IY+d)
FD3505	DEC	(IY+d)	FDCB05A6	RES	4,(IY+d)
FD360520	LD	(IY+d),n	FDCB05AE	RES	5,(IY+d)
FD39	ADD	IY,SP	FDCB05B6	RES	6,(IY+d)
FD4605	LD	B,(IY+d)	FDCB05BE	RES	7,(IY+d)
FD4E05	LD	C,(IY+d)	FDCB05C6	SET	0,(IY+d)
FD5605	LD	D,(IY+d)	FDCB05CE	SET	1,(IY+d)
FD5E05	LD	E,(IY+d)	FDCB05D6	SET	2,(IY+d)
FD6605	LD	H,(IY+d)	FDCB05DE	SET	3,(IY+d)
FD6E05	LD	L,(IY+d)	FDCB05E6	SET	4,(IY+d)
FD7005	LD	(IY+d),B	FDCB05EE	SET	5,(IY+d)
FD7105	LD	(IY+d),C	FDCB05F6	SET	6,(IY+d)

hex.code	mnemonic	
FDCB05FE	SET	7,(IY+d)
FDE1	POP	IY
FDE3	EX	(SP),IY
FDE5	PUSH	IY
FDE9	JP	(IY)
FDF9	LD	SP,IY
FE20	CP	n
FF	RST	38

24 Z80-vlagbeinvloeding

instructies	C	Z	P-V	S	N	H	opmerkingen
ADD A,s / ADC A,s	A	A	V	A	0	A	8 bits optelinstr.
SUB s / SBC A,s / CP s/ NEG	A	A	V	A	1	A	8 bits aftrekinstr., vergelijk- en negatie-instructies.
AND s	0	A	P	A	0	1	logische
OR s / XOR s	0	A	P	A	0	0	bewerkingen.
INC s	C	A	V	A	0	A	8 bits increment.
DEC s	C	A	V	A	1	A	8 bits decrement.
ADD dd,ss	A	C	C	C	0	X	16 bits optellen.
ADC HL,ss	A	A	V	A	0	X	16 bits optellen + CY
SBC HL,ss	A	A	V	A	1	X	16 bits aftrekken + CY
RLA/RLCA/RRA/RRCA	A	C	C	C	0	0	roteren accu.
RL s/ RLC s/ RR s RRC s/SLA s/SRA s SRL s	A	A	P	A	0	0	roteren en schuiven van locatie s.
RLD / RRD	C	A	P	A	0	0	roteren tetrade.
DAA	A	A	P	A	C	A	decimal adjust accu.
CPL	C	C	C	C	1	1	complementeren accu.
SCF	1	C	C	C	0	0	zet carry-vlag.
CCF	A	C	C	C	0	X	complementeer CY-vlag.
IN r,(C)	C	A	P	A	0	0	input in register.
INI/IND/OUTI/OUTD	C	A	X	X	1	X	blok-I/O. Indien B=0
INIR/INDR/OTIR/ OTDR	C	1	X	X	1	X	dan Z=1, anders Z=0.
LDI / LDD	C	X	A	X	0	0	blok-transfer. Als b=0
LDIR / LDDR	C	X	0	X	0	0	dan P/V=0, anders P/V=1.
CPI/CPIR/CPD/CPDR	C	A	A	A	1	X	blok-zoek. Als A=(HL) dan Z=1. Als BC=0 dan P/V=0.
LD A,I / LD A,R	C	A	F	A	0	0	IFF naar P/V-vlag.
BIT b,s	C	A	X	X	0	1	complement van bit b uit locatie s naar Z-vlag.

De betekenis van de in de voorgaande tabel gebruikte afkortingen wordt hieronder verklaard:

A - Vlag wordt afhankelijk van het resultaat van de bewerking gezet.
C - De vlag blijft onveranderd.
0 - De vlag wordt op 0 gezet.
1 - De vlag wordt op 1 gezet.
X - De status van de vlag is niet van belang.
V - De "overflow"-vlag wordt afhankelijk van het resultaat van de bewerking gezet.
P - De pariteitsvlag wordt afhankelijk van het resultaat van de bewerking gezet.
r - Een van de CPU-registers A, B, C, D, E, H of L.
s - Een 8 bits geheugenlocatie.
ss - Een 16 bits geheugenlocatie.
n - Een 8 bits waarde (0 t/m 255).
nn - Een 16 bits waarde (0 t/m 65535).

25 Systeemlocaties

In de hiernavolgende tabel worden een aantal systeem locaties opgesomd en van een korte omschrijving voorzien. Deze lijst is verre van compleet, doch de meeste bruikbare locaties zijn aanwezig.

ADRES Hex.	Dec.	Naam	Omschrijving
F3B0	62384	LINLEN	Aantal tekens per regel.
F3B1	62385	CRTCNT	Aantal regels op het scherm, vanaf de bovenkant.
F3B3/4	62387/8	TXTNAM	Startadres van de scherm info tabel in VRAM voor SCREEN 0.
F3B7/8	62391/2	TXTCGP	Startadres van de matrix tabel in VRAM voor SCREEN 0.
F3BD/E	62397/8	T32NAM	Startadres van de scherm info tabel in VRAM voor SCREEN 1.
F3BF/C0	62399/400	T32COL	Startadres van de kleur tabel in VRAM voor SCREEN 1.
F3C1/2	62401/2	T32CGP	Startadres van de matrix tabel in VRAM voor SCREEN 1.
F3C3/4	62403/4	T32ATR	Startadres van de sprite info tabel in VRAM voor SCREEN 1.
F3C5/6	62405/6	T32PAT	Startadres van de SPRITE$ tabel in VRAM voor SCREEN 1.
F3C7/8	62407/8	GRPNAM	Startadres van de scherm info tabel in VRAM voor SCREEN 2.
F3C9/A	62409/10	GRPCOL	Startadres van de kleur tabel in VRAM voor SCREEN 2.

ADRES Hex.	Dec.	Naam	Omschrijving
F3CB/C	62411/2	GRPCGP	Startadres van de matrix tabel in VRAM voor SCREEN 2.
F3CD/E	62413/4	GRPATR	Startadres van de sprite info tabel in VRAM voor SCREEN 2.
F3CF/D0	62415/6	GRPPAT	Startadres van de SPRITE$ tabel in VRAM voor SCREEN 2.
F3D1/2	62417/8	MLTNAM	Startadres van de scherm info tabel in VRAM voor SCREEN 3.
F3D5/6	62421/2	MLTCGP	Startadres van de matrix tabel in VRAM voor SCREEN 3.
F3D7/8	62423/4	MLTATR	Startadres van de sprite info tabel in VRAM voor SCREEN 3.
F3D9/A	62425/6	MLTPAT	Startadres van de SPRITE$ tabel in VRAM voor SCREEN 3.
F3DC	62428	CSRY	Verticale cursor positie.
F3DD	62429	CSRX	Horizontale cursor positie.
F3DE	62430	CNSDFG	Aan- en uitschakelen van het weergeven van de functietoetsen op schermregel 24.
F3DF	62431	RG0SAV	
F3E0	62432	RG1SAV	
F3E1	62433	RG2SAV	
F3E2	62434	RG3SAV	
F3E3	62435	RG4SAV	Copie van de inhoud van de besturingsregisters van de VDP.
F3E4	62436	RG5SAV	
F3E5	62437	RG6SAV	
F3E6	62438	RG7SAV	
F3E7	62439	STATFL	
F3E9	62441	FORCLR	Voorgrondkleur. Wordt actief na SCREEN-commando.
F3EA	62442	BAKCLR	Achtergrondkleur. Wordt actief na SCREEN-commando.

ADRES			
Hex.	Dec.	Naam	Omschrijving
F3EB	62443	BDRCLR	Randkleur. Wordt actief na SCREEN-commando.
F3F6	62454	SCNCNT	Tijdsinterval tussen twee key board scan-cycles.
F3F7	62455	REPCNT	Tijdsinterval voordat een ingedrukte toets begint te repeteren.
F3F8/9	62456/7	PUTPNT	Schrijf-pointer naar toetsenbordbuffer.
F3FA/B	62458/9	GETPNT	Lees-pointer naar toet-senbordbuffer.
F414	62484	ERRFLG	Fout nummer (saved).
F415	62485	LPTPOS	Positie van print-kop.
F416	62486	PRTFLG	0 = output naar scherm. 1 = output naar printer.
F417	62487	NTMSXP	0 = MSX-printer, anders non-MS-printer.
F55E/65F	62814/3071	TYPBUF	Invoer-buffer. Directe statements staan hier in.
F672/3	63090/1	MEMSIZ	Hoogste geheugenlocatie. (lsb/msb).
F674/5	63092/3	STKTOP	Hoogste locatie voor de stack (lsb/msb).
F69B/C	63131/2	FRETOP	Hoogste vrije locatie van string ruimte.
F6B3/4	63155/6	ERRLIN	Regelnummer van de regel waar de laatste fout in werd ontdekt.
F6B5/6	63157/8	DOT	Regelnummer van de regel die moet worden ge-edit of ge-list.
F6B9/A	63161/2	ONELIN	Regelnummer van de regel waarnaartoe moet worden gesprongen bij fouten.
F6BB	63163	ONEFLG	1 = foutafhandelings routine wordt uitge-voerd. (anders 0)
F85F	63583	MAXFIL	Maximum toegestane aantal tegelijkertijd geopende files.
F860/1	63584/5	FILTAB	Pointer naar tabel met adressen van file data.

ADRES Hex.	Dec.	Naam	Omschrijving
F862/3	63586/7	NULBUF	Pointer naar file 0 buffer.
F864/5	63588/9	PTRFIL	Pointer naar data van geselecteerde file.
F866/70	63590/600	FILNAM	File naam (eerste argument).
F871/B	63601/11	FILNM2	File naam (tweede argument).
F87F/91E	63615/774	FNKSTR	Teksten van de functie toetsen (10 * 16 tekens).
F922/3	63778/9	NAMBAS	Base-adres van de current scherm-info tabel.
F924/5	63780/1	CGPBAS	Base-adres van de current matrix-tabel.
F926/7	63782/3	PATBAS	Base-adres van de current SPRITE$-tabel.
F928/9	63784/5	ATRBAS	Base-adres van de current sprite-info tabel.
FC48/9	64584/5	BOTTOM	Laagste voor BASIC beschikbare RAM-adres.
FC4A/B	64586/7	HIMEM	Hoogste voor BASIC beschikbare RAM-adres.
FCAB	64683	CAPST	Status van de CAPS-toets. 0 = ON, 255 = OFF.
FCAF	64687	SCRMOD	Beeldschermmode.
FCBF/C0	64703/4	SAVENT	Startadres voor BSAVE, tevens executie-adres na BLOAD.
FCC1	64705	EXPSL1	&H80 = slot 1 expanded.
FCC2	64706	EXPSL2	&H80 = slot 2 expanded.
FCC3	64707	EXPSL3	&H80 = slot 3 expanded.
FCC4	64708	EXPSL4	&H80 = slot 4 expanded.
FCC5/8	64709/12	SLTTBL	1 byte voor ieder slot, waarin de waarde staat die naar het slot wordt uitgevoerd.
FCC9/D08	64713/86	SLTATR	1 byte per "page", met de attributes: Bit 7 = BASIC tekst. Bit 6 = Device expander. Bit 5 = Statement expander Bits 4-0 = niet gebruikt.

ADRES Hex.	Dec.	Naam	Omschrijving
FD09/88	64787/914	SLTWRK	Werkgebied van 2 bytes per "page".

Van de hiervoor opgesomde systeemlocaties volgt nu nog een alfabetische lijst op volgorde van de naam. Het achter de naam gegeven adres kan in de voorgaande opsomming gemakkelijk worden teruggevonden. Daarmee is dan ook de omschrijving teruggevonden.

Naam:	Adres:	Naam:	Adres:
ATRBAS	63784/5	LPTPOS	62485
BAKCLR	62442	MAXFIL	63583
BDRCLR	62443	MEMSIZ	63090/1
BOTTOM	64584/5	MLTATR	62423/4
CAPST	64683	MLTCGP	62421/2
CGPBAS	63780/1	MLTNAM	62417/8
CNSDFG	62430	MLTPAT	62425/6
CRTCNT	62385	NAMBAS	63778/9
CSRX	62429	NTMSXP	62487
CSRY	62428	NULBUF	63586/7
DOT	63157/8	ONEFLG	63163
ERRFLG	62484	ONELIN	63161/2
ERRLIN	63155/6	PATBAS	63782/3
EXPSL1	64705	PRTFLG	62486
EXPSL2	64706	PTRFIL	63588/9
EXPSL3	64707	PUTPNT	62456/7
EXPSL4	64708	REPCNT	62455
FILNAM	63590/600	RG0SAV	62431
FILNM2	63601/11	RG1SAV	62432
FILTAB	63584/5	RG2SAV	62433
FNKSTR	63615/774	RG3SAV	62434
FORCLR	62441	RG4SAV	62435
FRETOP	63131/2	RG5SAV	62436
GETPNT	62458/9	RG6SAV	62437
GRPATR	62413/4	RG7SAV	62438
GRPCGP	62411/2	SAVENT	64703/4
GRPCOL	62409/10	SCNCNT	62454
GRPNAM	62407/8	SCRMOD	64687
GRPPAT	62415/6	SLTATR	64713/86
HIMEM	64586/7	SLTTBL	64709/12
LINLEN	62384	SLTWRK	64787/914

Naam:	Adres:	Naam:	Adres:
STATFL	62439	T32NAM	62397/8
STKTOP	63092/3	T32PAT	62405/6
T32ATR	62403/4	TXTCGP	62391/2
T32CGP	62401/2	TXTNAM	62387/8
T32COL	62399/400	TYPBUF	62814/3071

26 Bios entry points

De volgende opsomming van BIOS entry points is niet volledig. Ik heb een selectie gemaakt van die punten, die ik ofwel zelf heb uitgetest, ofwel die mij nuttig lijken en die ik denk ooit nog eens te zullen gebruiken.

Het aanroepen van de BIOS entry points, direct vanuit BASIC, geschiedt als volgt:

```
DEFUSR=<adres van BIOS entry point>
DUMMY=USR(0)
```

Er worden dan geen gegevens in de Z80-registers geladen voordat het BIOS entry point wordt aangeroepen en er worden geen gegevens teruggegeven aan BASIC.

Heeft een BIOS entry point wel gegevens nodig, of geeft het gegevens terug, dan zal het aanroepen vanuit een machinetaal routinetje moeten geschieden. In die machinetaalroutine worden dan de registers van de Z80 geladen met de door de BIOS-routine vereiste gegevens. Deze gegevens kunnen al dan niet vanuit het BASIC-programma komen.

Ter verduidelijking twee voorbeeldjes. Met het eerste voorbeeld wordt BIOS entry point &H0000 aangeroepen. Dit entry point vereist geen input-gegevens en geeft geen gegevens terug. Het enige dat het aanroepen van dit entry point tot gevolg heeft is, dat de computer wordt ge-RESET. Gebruik dit voorbeeld daarom alleen, wanneer u niets in het geheugen hebt staan dat u wilt bewaren.

```
10 DEFUSR=0: REM 0 = &H0000
20 a=USR(0)
RUN
```

Het tweede voorbeeld laat zien, hoe een entry point, dat wel input-gegevens nodig heeft, vanuit een machinetaalroutine wordt aangeroepen.

```
10 CLEAR 500,&HD000: DEFUSR=&HD001
20 FOR I=1 TO 6
30 READ MC$: POKE &HD000+I,VAL("&H"+MC$)
40 NEXT I
50 DATA 3E,2A:'     * NAAR ACCU
60 DATA CD,A2,00:' CALL CHPUT
70 DATA C9:'       RETURN
80 CLS
90 FOR I=1 TO 100
100 A=USR(0)
110 NEXT I
120 END
```

In regel 50 staat de machinetaalinstructie, waarmee de Z80 accumulator wordt geladen met de code van een asterisk. Met de volgende machinetaalinstructie wordt BIOS entry point CHPUT aangeroepen. CHPUT verwacht, dat het af te drukken teken in de accumulator staat.

De beschrijving van de BIOS entry points in de hiernavolgende lijst bestaat steeds uit de volgende punten:

```
ADRES     NAAM
          FUNCTIE
          INPUT
          WIJZIGING
          OUTPUT
```

ADRES wordt als hexadecimaal adres gegeven. Achter het adres staat de naam van het BIOS entry point, zoals dat door Microsoft is gedefinieerd. Na de naam wordt een korte beschrijving van de functie van de BIOS routine gegeven.

INPUT geeft een opsomming van de registers die moeten worden geladen voordat de BIOS routine wordt aangeroepen.

WIJZIGT geeft aan welke registers door de BIOS routine zullen worden gewijzigd.

OUTPUT geeft een opsomming van de registers waarin de resultaten van de BIOS routine kunnen worden teruggevonden.

Indien een van de voornoemde punten niet van toepassing is, zal het betreffende punt uit de beschrijving worden weggelaten. Hierdoor kan de beschrijving zoveel mogelijk worden gecomprimeerd.

&H0000 CHKRAM

Start de initialisatie van de computer (soft-reset)

&H0008 RDSLT

Selecteert het in register A aangegeven slot en leest het geheugen in dit slot. Na uitvoering van deze routine zijn interrupts ge-disabled.

```
INPUT     A: FxxxSSPP
             |   ||||
             |   ||---- Primary Slot nummer
             |   ------ Secondary Slot nummer
             ---------- 0 = Primary Slot
                        1 = Secondary Slot
          HL: Geheugenadres
WIJZIGT   AF, BC, DE
OUTPUT    A : Inhoud van geheugenadres
```

&H0010 WRSLT

Selecteert het in register A aangegeven slot en schrijft naar het geheugen in dit slot. Na uitvoering van deze routine zijn interrupts ge-disabled.

```
INPUT     A : zie RDSLT
          HL: Geheugenadres
          E : Code van het te schrijven teken
WIJZIGT   AF, BC, D
```

&H001C CALSLT

Voert een inter-slot CALL uit naar het in IX aangegeven adres in het in IY aangegeven slot. Na uitvoering van deze routine zijn interrupts ge-disabled.

```
INPUT     IY(high): (zie RDSLT reg. A:)
          IX: het CALL-adres
```

&H0024 ENASLT

Selecteert het in register A aangegeven slot en activeert dat slot permanent. Na uitvoering van deze routine zijn interrupts gedisabled.

INPUT	A : zie RDSLT	
	HL: Geheugenadres	
WIJZIGT	Alle registers	

&H003B INITIO

Initialiseert de devices.

WIJZIGT Alle registers.

&H003E INIFNK

Initialiseert de functietoetsen (kent de default waarden toe aan de toetsen).

WIJZIGT Alle registers.

&H0041 DISSCR

Verhindert de schermweergave.

WIJZIGT AF, BC

&H0044 ENASCR

Laat schermweergave weer toe.

WIJZIGT AF, BC

&H0047 WRTVDP

Schrijft de opgegeven data naar het aangegeven VDP-register.

INPUT	B : te schrijven data
	C : te beschrijven VDP-register
WIJZIGT	AF, BC

&H004A RDVRM

Leest het in registerpaar HL aangegeven adres uit het video-RAM.

INPUT	HL: te lezen video-RAM-adres
WIJZIGT	AF
OUTPUT	A : de inhoud van het gelezen adres

&H004D WRTVRM

Schrijft de waarde uit register A naar het in registerpaar HL aangegeven video-RAM-adres.

INPUT	HL: te beschrijven video-RAM-adres
	A : te schrijven code
WIJZIGT	AF

&H0056 FILVRM
Vult het video-RAM met de in register A aangegeven data.

INPUT	HL:	VRAM-adres
	BC:	lengte
	A :	code van het teken
WIJZIGT	AF, BC	

&H0059 LDIRMV
Verplaatst een blok vanuit het video-RAM naar het geheugen.

INPUT	HL:	startadres van het blok in het video-RAM.
	DE:	startadres van het blok in het geheugen.
	BC:	lengte van het blok.
WIJZIGT	Alle registers	

&H005C LDIRVM
Verplaatst een blok vanuit het geheugen naar het video-RAM.

INPUT	HL:	startadres van het blok in het geheugen.
	DE:	startadres van het blok in het video-RAM.
	BC:	lengte van het blok.
WIJZIGT	Alle registers.	

&H005F CHGMOD
Zet de VDP in de mode, zoals die is aangegeven in systeemlocatie SCRMOD.

INPUT	SCRMOD (=adres &HFCAF)
WIJZIGT	Alle registers

&H0062 CHGCLR
Wijzigt de schermkleurinstelling.

INPUT	FORCLR (=adres &HF3E9)
	BAKCLR (=adres &HF3EA)
	BDRCLR (=adres &HF3EB)
WIJZIGT	Alle registers

&H0066 NMI
Voert een niet maskeerbare interrupt uit.

```
&H0084    CALPAT
          Levert  het startadres van de sprite patronen
          tabel op in registerpaar HL
          INPUT     A : Sprite nummer
          WIJZIGT   AF, DE, HL
          OUTPUT    HL: startadres  van sprite patronen
                        tabel.

&H0087    CALATR
          Levert  het startadres van de sprite attribu-
          tentabel op in registerpaar HL.
          INPUT     A : Sprite nummer.
          WIJZIGT   AF, DE, HL.
          OUTPUT    HL: startadres van de sprite attri-
                        butentabel.

&H008D    GRPPRT
          Drukt  een teken,  waarvan de code in de accu
          staat, af op het grafisch scherm.
          INPUT     A : Code  van  het  af  te  drukken
                        teken.

&H0090    GICINI
          Initialiseert de geluidsprocessor.
          WIJZIGT   Alle registers.

&H0093    WRTPSG
          Schrijft  gegevens  in  de registers  van  de
          geluidsprocessor (PSG).
          INPUT     A : nummer  van  het  PSG-register,
                        waarin moet worden geschreven.
                    E : waarde, die in het PSG-register
                        moet worden geschreven.

&H0096    RDPSG
          Leest de registers van de geluidsprocessor.
          INPUT     A : nummer  van  het  PSG-register,
                        dat moet worden gelezen.
          OUTPUT    E : de gelezen waarde.

&H009C    CHSNS
          Controleert of het keyboard-buffer leeg is.
          WIJZIGT   AF
          OUTPUT    Z-vlag: 0 is niet leeg.
                            1 is leeg.
```

&H009F CHGET
Wacht op invoer van het toetsenbord en geeft de code daarvan in register A.

WIJZIGT AF
OUTPUT A : code van het ingevoerde teken.

&H00A2 CHPUT
Drukt een teken af op het beeldscherm.

INPUT A : Code van het af te drukken teken.

&H00A5 LPTOUT
Drukt een teken af op de printer.

INPUT A : Code van het af te drukken teken.
WIJZIGT F
OUTPUT C-vlag: 0 is normaal verzonden.
1 is niet verzonden. (bijv. printer not ready)

&H00A8 LPTSTT
Controleert de status van de printer.

WIJZIGT AF
OUTPUT A : 0 is printer not ready.
255 is printer ready.
Z-vlag: 0 is printer not ready.
1 is printer ready.

&H00AE PINLIN
Leest invoer vanaf het toetsenbord in een buffer, totdat een CR of STOP toets is ingedrukt. Geeft als resultaat het startadres van het buffer.

WIJZIGT Alle registers.
OUTPUT HL: startadres van het buffer -1.
C-vlag: 0 is CR-toets
1 is STOP-toets

&H00B4 QINLIN
Schrijft een vraagteken en een spatie naar het beeldscherm en doet daarna exact hetzelfde als PINLIN.

WIJZIGT Alle registers.
OUTPUT HL: startadres van het buffer -1.
C-vlag: 0 is CR-toets
1 is STOP-toets

&H00C3 CLS
Wist het beeldscherm.
WIJZIGT AF, BC, DE.

&H00C6 POSIT
Positioneert de cursor op de aangegeven plaats.
INPUT H : Kolomnummer
L : Rijnummer
WIJZIGT AF

&H00CC ERAFNK
Wist de teksten van de functietoetsen van het scherm.
WIJZIGT Alle registers.

&H00CF DSPFNK
Schrijft de teksten van de functietoetsen op het scherm.
WIJZIGT Alle registers.

&H00D2 TOTEXT
Zet het beeldscherm in tekstmode.
WIJZIGT Alle registers.

&H00D5 GTSTCK
Geeft de status van de aangegeven joy-stick.
INPUT A : Joy-stick nummer.
WIJZIGT Alle registers.
OUTPUT A : Richtingnummer.

&H00D8 GTTRIG
Geeft de status van de aangegeven actieknop.
INPUT A : Actieknopnummer.
WIJZIGT AF
OUTPUT A : 0 is niet ingedrukt.
255 is wel ingedrukt.

&H00E1 TAPION
Zet de cassetterecordermotor aan en leest een header van de cassette.
WIJZIGT Alle registers.
OUTPUT C-vlag: 0 is normaal gelezen.
1 is lezen afgebroken.
(Zie ook diverse systeemlocaties.)

```
&H00E4    TAPIN
          Leest gegevens van cassette.
          WIJZIGT   Alle registers.
          OUTPUT    A : Het gelezen byte.
                    C-vlag: 0 is normaal gelezen.
                            1 is lezen onderbroken.

&H00E7    TAPIOF
          Stopt het lezen van de cassette.

&H00EA    TAPOON
          Start  de  cassetterecordermotor en  schrijft
          de header naar cassette.
          INPUT     A : 0 is korte header
                        1 is lange header
          WIJZIGT   Alle registers
          OUTPUT    C-vlag: 0 is normaal geschreven.
                            1 is schrijven onderbroken.

&H00ED    TAPOUT
          Schrijft naar cassette.
          INPUT     A : te schrijven code.
          WIJZIGT   Alle registers.
          OUTPUT    C-vlag: 0 is normaal geschreven.
                            1 is schrijven onderbroken.

&H00F0    TAPOOF
          Stopt het schrijven naar cassette.

&H00F3    STMOTR
          Cassetterecordermotor  besturing.
          INPUT     A : 0   is stop de motor.
                        1   is start de motor.
                        255 start de motor indien  deze
                            stil  staat  en  stopt  de
                            motor indien deze draait.
          WIJZIGT   AF

&H00FC    RIGHTC
          Verplaatst  de cursor op het grafisch  scherm
          een pixel naar rechts.

&H00FF    LEFTC
          Verplaatst  de cursor op het grafisch  scherm
          een pixel naar links.
```

&H0102 UPC
Verplaatst de cursor op het grafisch scherm een pixel omhoog.

&H0108 DOWNC
Verplaatst de cursor op het grafisch scherm een pixel naar beneden.

&H0132 CHGCAP
Wijzigt de status van de hoofdletter-indicatielamp.

INPUT	A :	0 is lamp uit. 1 is lamp aan.
WIJZIGT	AF	

&H013E RDVDP
Leest het statusregister van de videoprocessor.

WIJZIGT	A	
OUTPUT	A :	de inhoud van het statusregister.

&H0141 SNSMAT
Leest de stand van de schakelaartjes van een gegeven rij van het toetsenbord.

INPUT	A :	Rijnummer
WIJZIGT	AF	
OUTPUT	A :	Binaire waarde van de 8 schakelaartjes. Bitwaarde 0 wil zeggen: toets ingedrukt.

&H0156 KILBUF
Wist het toetsenbordbuffer.
WIJZIGT HL

Quick reference lijst in alfabetische volgorde van de BIOS entry point namen.

In deze lijst staan alleen de namen van de BIOS entry points die in de voorgaande beschrijving zijn opgenomen.

NAAM	ADRES	NAAM	ADRES
CALATR	&H0087	LEFTC	&H00FF
CALPAT	&H0084	LPTOUT	&H00A5
CALSLT	&H001C	LPTSTT	&H00A8
CHGCAP	&H0132	NMI	&H0066
CHGCLR	&H0062	PINLIN	&H00AE
CHGET	&H009F	POSIT	&H00C6
CHGMOD	&H005F	QINLIN	&H00B4
CHKRAM	&H0000	RDPSG	&H0096
CHPUT	&H00A2	RDSLT	&H0008
CHSNS	&H009C	RDVDP	&H013E
CLS	&H00C3	RDVRM	&H004A
DISSCR	&H0041	RIGHTC	&H00FC
DOWNC	&H0108	SNSMAT	&H0141
DSPFNK	&H00CF	STMOTR	&H00F3
ENASCR	&H0044	TAPIN	&H00E4
ENASLT	&H0024	TAPIOF	&H00E7
ERAFNK	&H00CC	TAPION	&H00E1
FILVRM	&H0056	TAPOOF	&H00F0
GICINI	&H0090	TAPOON	&H00EA
GRPPRT	&H008D	TAPOUT	&H00ED
GTSTCK	&H00D5	TOTEXT	&H00D2
GTTRIG	&H00D8	UPC	&H0102
INIFNK	&H003E	WRSLT	&H0010
INITIO	&H003B	WRTPSG	&H0093
KILBUF	&H0156	WRTVDP	&H0047
LDIRMV	&H0059	WRTVRM	&H004D
LDIRVM	&H005C		

27 Hook-adressen

Het systeemdeel van het RAM-geheugen vanaf adres &HFD9A tot en met adres &HFFCA is gereserveerd voor zogenaamde "hook"-adressen. Dit wil zeggen, dat er vanuit de BIOS-routines naar een adres in dit "hook"-gebied wordt gesprongen. Normaal staan er in het "hook"-gebied alleen machinetaal-RETURN-instructies. Er wordt dan direct teruggesprongen naar de BIOS-routine, zonder dat er iets is gebeurd. Als wij echter op een "hook"-adres een korte machinetaalroutine zetten (er zijn vijf geheugenadressen beschikbaar), of indien we op dat "hook"-adres een sproninginstructie plaatsen naar een machinetaalroutine elders in het geheugen, dat kan de BIOS-routine als het ware worden uitgebreid of vervangen met of door een eigen machinetaalroutine.

In de hiernavolgende beschrijving zal van ieder "hook"-adres het adres, de naam en een korte omschrijving worden gegeven. Een aantal "hook"-adressen hebben geen naam, of een mij niet duidelijke functie. Deze adressen heb ik uit de lijst weggelaten.

&HFD9A KEYI
Wordt aangeroepen aan het begin van de interrupt afhandelingsroutine en dient voor interrupt-afhandeling van niet standaard interrupts (bijvoorbeeld RS232).

&HFD9F TIMI
Wordt aangeroepen in de timer interrupt routine en maakt extra timer-interrupt afhankelijke interrupt-afhandeling mogelijk.

&HFDA4 CHPU
Wordt aangeroepen aan het begin van de CHaracter outPUT routine, om andere output devices dan het beeldscherm mogelijk te maken.

&HFDA9 DSPC
Wordt aangeroepen aan het begin van de DiSPlay Cursor routine, om andere output devices dan het beeldscherm mogelijk te maken.

&HFDAE ERAC
Wordt aangeroepen aan het begin van de ERAse Cursor routine, om andere output devices dan het beeldscherm mogelijk te maken.

&HFDB3 DSPF
Wordt aangeroepen aan het begin van de DiSPlay Function key routine, om andere output devices dan het beeldscherm mogelijk te maken.

&HFDB8 ERAF
Wordt aangeroepen aan het begin van de ERAse Function key routine, om andere output devices dan het beeldscherm mogelijk te maken.

&HFDBD TOTE
Wordt aangeroepen aan het begin van de force screen TO TExt mode routine, om andere output devices dan het beeldscherm mogelijk te maken.

&HFDC2 CHGE
Wordt aangeroepen aan het begin van de CHaracter GEt routine, om andere input devices dan het toetsenbord mogelijk te maken.

&HFDC7 INIP
Wordt aangeroepen aan het begin van de INItialize Pattern routine, om andere tekensets mogelijk te maken.

&HFDCC KEYC
Wordt aangeroepen aan het begin van de KEY Coder routine, om andere tekens aan toetsen toe te kennen.

&HFDD1 KYEA
Wordt aangeroepen aan het begin van de KeY EAsy routine, om andere tekens aan toetsen toe te kennen.

&HFDD6 NMI
Wordt aan het begin van de Non Maskable Interrupt routine aangeroepen, om Non Maskable Interrupt afhandeling mogelijk te maken.

&HFDDB PINL
Wordt aangeroepen aan het begin van de Program INput Line routine, om andere input devices dan toetsenbord mogelijk te maken.

&HFDE0 QINL
Wordt aangeroepen aan het begin van de Question mark and INput Line routine om andere input devices dan het toetsenbord mogelijk te maken.

&HFDE5 INLI
Wordt aangeroepen aan het begin van de INput LIne routine, om andere input devices dan het toetsenbord mogelijk te maken.

&HFDEA ONGO
Wordt aangeroepen aan het begin van de ON GOto procedure routine, om andere input devices dan het toetsenbord mogelijk te maken.

&HFDEF DSKO
Wordt aangeroepen aan het begin van de DiSK Output routine, om een disk driver te installeren.

&HFDF4 SETS
Wordt aangeroepen aan het begin van de SET attributeS routine, om een disk driver te installeren.

&HFDF9 NAME
Wordt aangeroepen aan het begin van de reNAME routine, om een disk driver te installeren.

&HFDFE KILL
Wordt aangeroepen aan het begin van de KILL file routine, om een disk driver te installeren.

&HFE03 IPL
Wordt aangeroepen aan het begin van de Initial Program Load routine, om een disk driver te installeren.

&HFE08 COPY
Wordt aangeroepen aan het begin van de COPY file routine, om een disk driver te installeren.

&HFE0D CMD
Wordt aangeroepen aan het begin van de CoMmanD routine, om een disk driver te installeren.

&HFE12 DSKF
Wordt aangeroepen aan het begin van de DiSK Free routine, om een disk driver te installeren.

&HFE17 DSKI
Wordt aangeroepen aan het begin van de DiSK Input routine, om een disk driver te installeren.

&HFE1C ATTR
Wordt aangeroepen aan het begin van de ATTRibute routine, om een disk driver te installeren.

&HFE21 LSET
Wordt aangeroepen aan het begin van de Left SET routine, om een disk driver te installeren.

&HFE26 RSET
Wordt aangeroepen aan het begin van de Right SET routine, om een disk driver te installeren.

&HFE2B FIEL
Wordt aangeroepen aan het begin van de FIELd routine, om een disk driver te installeren.

&HFE30 MKI
Wordt aangeroepen aan het begin van de MaKe Int routine, om een disk driver te installeren.

&HFE35 MKS
Wordt aangeroepen aan het begin van de MaKe Single routine, om een disk driver te installeren.

&HFE3A MKD
Wordt aangeroepen aan het begin van de MaKe Double routine, om een disk driver te installeren.

&HFE3F CVI
Wordt aangeroepen aan het begin van de ConVert Integer routine, om een disk driver te installeren.

&HFE44 CVS
Wordt aangeroepen aan het begin van de ConVert Single routine, om een disk driver te installeren.

&HFE49 CVD
Wordt aangeroepen aan het begin van de ConVert Double routine, om een disk driver te installeren.

&HFE4E GETP
Wordt aangeroepen aan het begin van de GET file Pointer routine, om een disk driver te installeren.

&HFE53 SETF
Wordt aangeroepen aan het begin van de SET File pointer routine, om een disk driver te installeren.

&HFE58 NOFO
Wordt aangeroepen aan het begin van de NO FOR clause routine, om een disk driver te installeren.

&HFE5D NULO
Wordt aangeroepen aan het begin van de NULl file Open routine, om een disk driver te installeren.

&HFE62 NTFL
Wordt aangeroepen aan het begin van de NoT FiLe number 0 routine, om een disk driver te installeren.

&HFE67 MERG
Wordt aangeroepen aan het begin van de MERGe program files routine, om een disk driver te installeren.

&HFE6C SAVE
Wordt aangeroepen aan het begin van de SAVE routine, om een disk driver te installeren.

&HFE71 BINS
Wordt aangeroepen aan het begin van de BINary Save routine, om een disk driver te installeren.

&HFE76 BINL
Wordt aangeroepen aan het begin van de BINary Load routine, om een disk driver te installeren.

&HFE7B FILE
Wordt aangeroepen aan het begin van de FILEs routine, om een disk driver te installeren.

&HFE80 DGET
Wordt aangeroepen aan het begin van de Disk GET routine om een disk driver te installeren.

&HFE85 FILO
Wordt aangeroepen aan het begin van de FILe Out routine om een disk driver te installeren.

&HFE8A INDS
Wordt aangeroepen aan het begin van de INput DiSk routine, om een disk driver te installeren.

&HFE8F RSLF
Wordt aangeroepen aan het begin van de Re-SeLect old drive routine, om een disk driver te installeren.

&HFE94 SAVD
Wordt aangeroepen aan het begin van de SAVe current Drive routine, om een disk driver te installeren.

&HFE99 LOC
Wordt aangeroepen aan het begin van de LOCation routine, om een disk driver te installeren.

&HFE9E LOF
Wordt aangeroepen aan het begin van de Length Of File routine, om een disk driver te installeren.

&HFEA3 EOF
Wordt aangeroepen aan het begin van de End Of File routine, om een disk driver te installeren.

&HFEA8 FPOS
Wordt aangeroepen aan het begin van de File POSition routine, om een disk driver te installeren.

&HFEAD BAKU
Wordt aangeroepen aan het begin van de BAcK Up routine, om een disk driver te installeren.

&HFEB2 PARD
Wordt aangeroepen aan het begin van de PARse Device name routine, om het aantal logische device namen te kunnen uitbreiden.

&HFEB7 NODE
Wordt aangeroepen aan het begin van de NO DEvice name routine, om andere default devices te kunnen definieren

&HFEBC POSD
Wordt aangeroepen aan het begin van de POSsibly Disk routine, om een disk driver te installeren.

&HFEC1 DEVN
Wordt aangeroepen aan het begin van de DEVice Name routine, om het aantal logische device namen te kunnen uitbreiden.

&HFEC6 GEND
Wordt aangeroepen aan het begin van de GENeral Device dispatcher routine, om het aantal logische device namen te kunnen uitbreiden.

&HFED5 LOPD
Wordt aangeroepen aan het begin van de LOoP and set Default routine, om andere default waarden aan variabelen toe te kennen.

&HFF89 LIST
Wordt aangeroepen aan het begin van de LIST routine, om listings te beinvloeden of voorkomen.

&HFFA7 PHYD
Wordt aangeroepen aan het begin van de PHYsical Disk routine, om een disk driver te installeren.

&HFFAC FORM
Wordt aangeroepen aan het begin van de FORMat disk routine, om een disk driver te installeren.

&HFFB1 ERRO
Wordt aangeroepen aan het begin van de ERROr routine, om programmafouten in een eigen foutafhandelingsroutine te kunnen afhandelen.

&HFFB6 LPTO
Wordt aangeroepen aan het begin van de Line PrinTer Output routine, om een printer driver te installeren.

&HFFBB LPTS
Wordt aangeroepen aan het begin van de Line PrinTer Status routine, om een printer driver te installeren.

&HFFC0   SCRE
Wordt aangeroepen aan het begin van de SCREEN-routine, om het SCREEN statement te kunnen uitbreiden.

&HFFC5   PLAY
Wordt aangeroepen aan het begin van de PLAY routine, om het PLAY-statement te kunnen uitbreiden.

28 Indeling van MSXDOS-schijven

Iedere schijf is opgedeeld in een aantal sporen (tracks). Iedere track is weer opgedeeld in een aantal sectoren. De sectoren worden genummerd van 0 tot en met de laatste. Daar het aantal sporen en het aantal sectoren per spoor kan varieren, is het niet mogelijk een vast nummer te geven voor de laatste sector op de schijf.

De "boot"-sector.

Alle schijven hebben gemeen, dat de eerste sector op de schijf (sector 0) de zogenaamde "boot"-sector is. Deze "boot"-sector heeft een speciale indeling, die er als volgt uitziet:

Byte:	Betekenis:
&H00	- &HEB of $HE9 Het eerste byte van de eerste sector moet een van deze waarden hebben als het een MSXDOS-schijf is. In dat geval vindt u op displacements &H0B tot en met &H1D het Drive Parameter Block (DPB).
&H03/0A	- OEM-naam van de computer fabrikant. Deze ruimte wordt voor MS-DOS floppies gebruikt om er een Volume naam in te zetten (zie ook het attribute byte).
&H0B/0C	- Aantal bytes per sector. &H0B is het minst significante byte.
&H0D	- Aantal sectoren in een cluster.
&H0E/0F	- Aantal gereserveerde sectoren. &H0E is het minst significante byte.
&H10	- Aantal File Allocation Tabellen (FAT).

Byte:	Betekenis:
&H11/12 -	Aantal directory entries. &H11 is het minst significante byte.
&H13/14 -	Het totaal aantal sectoren op de schijf. &H13 is het minst significante byte.
&H15 -	Medium descriptor: &HF8 = 80 tracks, 9 sectoren/track, 1 side &HF9 = 80 tracks, 9 sectoren/track, 2 sides &HFA = 80 tracks, 8 sectoren/track, 1 side &HFB = 80 tracks, 8 sectoren/track, 2 sides &HFC = 40 tracks, 9 sectoren/track, 1 side &HFD = 40 tracks, 9 sectoren/track, 2 sides &HFE = 40 tracks, 8 sectoren/track, 1 side &HFF = 40 tracks, 8 sectoren/track, 2 sides
&H16/17 -	Aantal sectoren per FAT. &H16 is het minst significante byte.
&H18/19 -	Aantal sectoren per track. &H18 is het minst significante byte.
&H1A/1B -	Aantal zijden. &H1A is het minst significante byte. Het aantal zijden kan 1 of twee zijn voor floppy disk. Het "boot"-sectorformaat wordt echter ook voor harde schijven gebruikt, en daar kan een groter aantal zijden aanwezig zijn.
&H1C/1D -	Aantal verborgen sectoren. &H1C is het minst significante byte.

De rest van sector 0 bevat een "boot"-routine en een aantal ongebruikte bytes.

De File Allocation Tabel.

De File Allocation Tabel (FAT) start op sector 1. De lengte (aantal sectoren) is afhankelijk van het medium-type. De FAT is opgedeeld in groepjes van 1,5 byte, ofwel groepjes van 12 bits. De groepjes kunnen de volgende inhoud hebben:

Inh.: Betekenis:

000 - Het cluster is vrij (wordt niet gebruikt).

FF7 - Het cluster bevat een "bad" sector. Het betreffende cluster zal door MSXDOS niet worden toegekend aan een file.

FF8 - FF8 of hoger geeft het laatste aan een file toegekende cluster aan.

xxx - Elke andere waarde dan een hiervoor genoemde waarde geeft het nummer van het volgende cluster in de file aan.

Om de FAT te kunnen lezen dient u de volgende procedure aan te houden.

- Vermenigvuldig het laatst gebruikte cluster nummer met 1.5. Het gehele deel van het resultaat van de vermenigvuldiging is een pointer.
- Gebruik de pointer om de FAT-entry van het in gebruik zijnde cluster te lezen. In deze entry vindt u het cluster nummer van het volgende cluster dat door de file wordt gebruikt (zie hierna).
- Lees de drie bytes vanaf het byte dat met de pointer wordt aangewezen.
- Indien het laatst gelezen cluster een even nummer had, dan bevatten de 12 minst significante bits van de drie bytes het nummer van het volgende cluster in de file. Had de laatst gelezen cluster een oneven nummer, dan bevatten de 12 meest significante bits van de drie bytes het nummer van het volgende cluster in de file.
- Trek 2 af van het gevonden cluster nummer (het volgende cluster nummer in de file).
- Vermenigvuldig het resultaat van de aftrekking met het aantal sectoren per cluster.
- Tel de gevonden waarde op bij het sector nummer van de eerste data sector en u hebt het sector nummer, waarop het betreffende cluster (het volgende cluster in de file) begint, gevonden.

De directory.

Na de FAT begint de directory. Ook de sectoren van de directory hebben een vaste indeling. Iedere sector van de directory bevat 8 entries, die allemaal dezelfde onderverdeling hebben. Hierna volgt een beschrijving van 1 directory entry:

Displ.:	Betekenis:
&H00/07 -	De file-naam. Maximaal 8 tekens. Indien de naam uit minder dan 8 tekens bestaat, zullen de overblijvende plaatsen met spaties (&H20) worden gevuld. Behalve een file-naam zijn er nog de volgende mogelijkheden: Entry is nog nooit gebruikt geweest: Displacement &H00 bevat de waarde &H00. Entry bevat een (sub)directory: Displacement &H00 bevat de waarde &H2E. Entry heeft een gewiste file bevat: Displacement &H00 bevat de waarde &HE5.
&H08/0A -	File-naam uitbreiding. Maximaal 3 tekens. Indien de uitbreiding uit minder dan 3 tekens bestaat, zullen de overblijvende plaatsen met spaties (&H20) worden gevuld.
&H0B -	Attribuut Byte. Dit byte kan de volgende waarden bevatten (in MSX bevat dit byte default de waarde 0): &H00 - Normale file. &H01 - Read-only file. &H02 - Verborgen file (niet zichtbaar in de directory). &H04 - Systeem file (niet zichtbaar in de directory). &H08 - De eerste 11 bytes bevatten het Volume label. &H10 - Deze entry betreft een sub-directory. &H20 - Archive bit (wordt ge-set wanneer er naar de file is geschreven). Geen van de attributen wordt in MSXDOS gebruikt (behalve &H00), doch het is wel mogelijk om de attributen zelf te wijzigen.

Displ.:	Betekenis:
	Zou men bijvoorbeeld het attribuut voor een bepaalde file &H02 maken, dan zal die file niet meer in de directory voorkomen. Hij is dan werkelijk verborgen.
&H0C/15 -	Gereserveerd.
&H16/17 -	Creatie-tijd van de file. De twee bytes hebben de volgende indeling: &H17: H H H H H M M M &H16: M M M S S S S S Hierin is: H = 0 - 23 (binaire waarde voor uren). M = 0 - 59 (binaire waarde voor minuten). S = 0 - 29 (binaire waarde voor aantal perioden van 2 seconden).
&H18/19 -	Creatie-datum van de file. De twee bytes hebben de volgende indeling: &H19: J J J J J J J M &H18: M M M D D D D D Hierin is: J = 0 - 119 (binaire waarde voor het jaar). (0 = 1980, 119 = 2099). M = 1 - 12 (binaire waarde voor de maand). D = 1 - 31 (binaire waarde voor de dag).
&H1A/1B -	Het nummer van het eerste cluster in de file. &H1A is het minst significante byte. Het cluster nummer van de eerste file op iedere schijf is altijd nummer 2.
&H1C/1F -	File-grootte in bytes. &H1C is het minst significante byte van deze groep van 4 bytes.

29 File Control Block voor disk BASIC

Disk-BASIC maakt, voor het correct werken met files op schijven, gebruik van een aantal administratiegebieden in het RAM-geheugen. Deze administratiegebieden worden File Control Blocks genoemd, kortweg aangeduid met FCB.

Voor iedere file die wordt geopend, wordt zo'n File Control Block aangelegd. Het bij een file behorende FCB kan worden teruggevonden door gebruik te maken van de VARPTR-functie.

Indien een disk-file werd geopend met het volgende statement:

```
OPEN "A:FILENAAM" FOR OUTPUT AS #1
```

dan zal de functie:

```
VARPTR(#1)
```

het start-adres van het File Control Block opleveren. Dit File Control Block heeft de volgende layout:

displ.:	Veld:	Betekenis:
&H00	MOD	Mode waarin de file werd geopend.
&H01/02	FCA	Pointer naar het DOS-FCB. (lsb/msb)
&H03	LSA	Back up teken.
&H04	DSK	Device-nummer.
&H05	SLB	?
&H06	BPS	Positie binnen het data-buffer.
&H07	FLG	Flag-info.
&H08	OPS	Pseudo kop-positie.
&H09/108	BUF	Data buffer. (256 bytes)

Het data-buffer bevat de informatie die naar schijf moet worden geschreven of die van schijf is gelezen.

De pointer naar het DOS-FCB wijst naar een uitgebreider File Control Block, waarin gegevens zoals de file-naam, de plaats binnen de directory, de voor de file gebruikte sectoren, de tijd en de datum, etc. kunnen worden gevonden. De layout van die FCB wordt gegeven in het hoofdstuk waarin de BDOS-calls worden behandeld.

30 BDOS Calls

Wanneer aan de MSX-computer een disk drive met controller is aangesloten, dan bevinden zich alle disk-routines zich (net als de Basic IO System routines) in het geheugen van de computer. Ook voor deze disk routines geldt, dat ze vanuit een applicatieprogramma kunnen worden aangeroepen. Om te voorkomen, dat een programma wel op de ene maar niet op een andere computer werkt, moeten ook deze disk routines (BDOS-routines) door middel van een speciale manier van aanroepen worden gebruikt.

De manier om deze routines aan te roepen is als volgt:

1 - Laadt het functienummer van de aan te roepen routine in register C van de Z80-microprocessor.

2 - CALL het adres &HF37D, indien het aanroepende programma onder Disk BASIC draait.
CALL het adres &H0005, indien het aanroepende programma onder MSXDOS draait.

Daar een groot aantal BDOS calls gebruik maken van het File Control Block (FCB) en van het Drive Parameter Block (DPB) zullen de indelingen van deze blokken worden gegeven, voordat op de BDOS calls zelf wordt ingegaan.

FILE CONTROL BLOCK (37 bytes):

Displ.:	Betekenis:
&H00	Drive naam (default = 0, A: = 1)
&H01/08	De file-naam. Maximaal 8 tekens. Indien de naam uit minder dan 8 tekens bestaat, dienen de overgebleven posities uit spaties te bestaand.
&H09/0B	De file-naam extentie. Maximaal 3 tekens. Indien de extentie uit minder dan 3 tekens bestaat, dienen de overgebleven posities uit spaties te bestaan.

&H0C/0D	Current Block. Pointer naar het blok waarin het record dat zal worden gelezen zich bevindt.
&H0E/0F	Record Size. Default 128. Kan een andere waarde worden gegeven, doch niet langer dan 255.
&H10/13	File Size. Grootte van de file in aantal blokken van 128 bytes.
&H14/15	Datum.
&H16/17	Tijd.
&H18	Device ID.
&H19	Directory Location.
&H1A/1B	First Cluster van de file.
&H1C/1D	Last Cluster Accessed.
&H1E/1F	Last Cluster Accessed. Relatief ten opzichte van het eerste cluster van de file.
&H20	Current Record.
&H21/24	Random Record. Indien de record lengte korter is dan 64 bytes, worden alleen de displacements &H21/23 gebruikt.

DRIVE PARAMETER BLOCK

Displ.:	Betekenis:
&H00	Drive nummer
&H01	Media ID byte.
&H02/03	Aantal bytes per sector.
&H04	Directory masker.
&H05	Directory shift.
&H06	Cluster masker.
&H07	Cluster shift.
&H08/09	Eerste sector van de FAT.
&H0A	Aantal FAT's.
&H0B	Aantal directory entries.
&H0C/0D	Eerste data sector.
&H0E/0F	Aantal clusters + 1
&H10	Aantal sectoren per FAT.
&H11/12	Eerste directory sector.
&H13/14	FAT-adres.

In de hierna volgende opsomming van BDOS calls worden van iedere Call de volgende gegevens opgenomen:

NUMMER
NAAM
FUNCTIE
INPUT
OUTPUT

NUMMER is het functienummer, dat in register C moet worden geplaatst voordat adres &HF37D of &H0005 wordt aangeroepen.

NAAM is de naam van de aan te roepen functie.

FUNCTIE is een korte uitleg van wat de functie doet.

INPUT geeft een opsomming van de register die moeten worden geladen met gegevens die de functie nodig heeft om het gewenste resultaat op te leveren.
OUTPUT geeft een opsomming van de registers die het resultaat van de functie bevatten.

Indien een van de voornoemde punten niet van toepassing is, zal het betreffende punt uit de beschrijving worden weggelaten.

&H00 SYSTEM RESET
Springt naar de "warme start" van Disk BASIC of naar adres &H0000 indien de functie onder MSXDOS werd aangeroepen. CP/M-compatibel.

&H01 CONSOLE INPUT
Leest een teken van console en zet dit in register A. Voert functie &H00 uit indien CONTROL+C werd ingedrukt. Start output van beeldscherminfo naar de printer wanneer CONTROL+P werd ingedrukt en beeindigt dit weer indien CONTROL+N werd ingedrukt. Stuurt het ingevoerde teken naar het beeldscherm. CP/M-compatibel.
OUTPUT A: Het ingetoetste teken.

&H02 CONSOLE OUTPUT
Stuurt het teken dat in register E staat naar de console. CP/M-compatibel.
INPUT E: Het naar console te schrijven teken.

&H03 AUX INPUT
Leest een teken van het AUX-device (Bijv. RS232) en zet dit in register A. CP/M-compatibel.
OUTPUT A: Het gelezen teken.

&H04 AUX OUTPUT
Schrijft het teken uit register E naar het AUX-device. CP/M-compatibel.
INPUT E: Het naar AUX te schrijven teken.

&H05 LST OUTPUT
Schrijft het teken uit register E naar de printer. CP/M-compatibel.
INPUT E: Het af te drukken teken.

&H06 DIRECT CONSOLE IO
Stuurt het teken dat in register E staat naar de console, tenzij in register E de code &HFF staat. In dat geval wordt van het console gelezen. Het gelezen teken wordt in register A gezet, zonder dat er op de ingetoetste code wordt gechecked en zonder dat het ingetoetste teken naar console wordt gestuurd. Indien er niets wordt ingetoetst, zal register A de waarde 0 bevatten. CP/M-compatibel.
INPUT E: &HFF indien moet worden gelezen.
Iedere andere code wordt naar console geschreven.
OUTPUT A: Het ingelezen teken.

&H07 DIRECT INPUT
Leest een teken van console en zet dat in register A. Er wordt niet op de code van het teken gechecked en het teken wordt niet naar console teruggestuurd.
OUTPUT A: Het ingelezen teken.

&H08 DIRECT INPUT

Leest een teken van console en zet dat in register A. Controleert of CONTROL+C, CONTROL+P of CONTROL+N is ingedrukt. Stuurt het ingetoetste teken niet terug naar console.

OUTPUT   A: Het ingetoetste teken.

&H09 STRING OUTPUT

Schrijft de string, waarvan het startadres in registerpaar DE staat, naar console. Het schrijven gaat door, totdat een $-teken wordt gevonden. CP/M-compatibel.

INPUT   DE: Het startadres van de string.

&H0A BUFFERED INPUT

Leest een string van console en zet deze in het geheugen vanaf het in registerpaar DE gegeven adres+2. CR beeeindigt de string. De lengte van de string wordt op het in registerpaar DE gegeven adres+1 gezet. CR wordt niet meegeteld in de lengte. De maximale lengte van de string wordt doorgegeven via de geheugenlocatie die met het adres in DE wordt aangewezen. CP/M-compatibel.

INPUT   DE: Pointer naar string-ruimte.

&H0B CONSOLE STATUS

Controleert of er input van console wordt gedaan. Zet 0 in register A indien er geen input is. Zet &HFF in register A indien er wel input is. CP/M-compatibel.

OUTPUT   A: Console status.

&H0C GET VERSION NUMBER

Zet het versie nummer in registerpaar HL. CP/M-compatibel.

OUTPUT   H: &H00
          L: &H22

&H0D DISK RESET

Stelt drive A: in als default drive, zet het transfer-adres op &H80 en verwijdert alle sectoren in het geheugen die zijn gewijzigd, maar nog niet naar schijf zijn geschreven. CP/M-compatibel.

&H0E SELECT DISK

Maakt de in register E gegeven disk de default drive (0 komt overeen met drive A:). CP/M-compatibel.

INPUT   E: drive nummer.

&H0F OPEN FILE

Opent een file volgens de specificaties in het File Control Block waar met het adres in registerpaar DE naartoe wordt verwezen. De volgende velden van het FCB worden gevuld vanuit de directory:

- File Size
- Date
- Time
- Device ID
- Directory Location
- First Cluster
- Last Cluster
- Last Accessed Cluster.

De volgende velden van het FCB zijn ge-set na uitvoering van deze functie:

- Record Size
- Current Block
- Current Record
- Random Record

Indien de functie succesvol is uitgevoerd zal register A de waarde &H00 bevatten, zoniet, dan zal register A de waarde &HFF bevatten. CP/M-compatibel.

```
INPUT     DE: FCB-adres
OUTPUT    A : &H00 = succesvol
              &HFF = niet succesvol
```

&H10 CLOSE FILE

Sluit de file die in het File Control Block staat dat met het adres in registerpaar DE wordt aangewezen. Indien de file succesvol is gesloten zal register A de waarde &H00 bevatten, zoniet dan bevat register A de waarde &HFF. CP/M-compatibel.

```
INPUT     DE: FCB-adres.
OUTPUT    A : &H00 = succesvol
              &HFF = niet succesvol
```

&H11 SEARCH FIRST

Zoekt vanaf het begin van de directory naar het voorkomen van de file, die in het File Control Block dat met het adres in registerpaar DE wordt aangewezen staat aangegeven, in de directory en copieert de directory-entry (32 bytes lang) naar het transfer-adres. Indien de file in de directory werd gevonden zal bovendien register A met &H00 worden geladen. Wordt de file niet gevonden, dan wordt niets naar het transfer-adres gecopieerd en wordt register A met de waarde &HFF geladen. In de file-naam (in het File Control Block mag gebruik worden gemaakt van de tekens * en ?. CP/M-compatibel.

INPUT   DE: FCB-adres.
OUTPUT   A : &H00 = File gevonden.
              &HFF = File niet gevonden.

&H12 SEARCH NEXT

Zoekt in de directory vanaf de plaats waar met de laatste SEARCH-functie een gezochte file werd gevonden naar het volgende voorkomen van die file-naam in de directory. Werkt voor het overige precies zoals de SEARCH FIRST functie. CP/M-compatibel.

INPUT   DE: FCB-adres.
OUTPUT   A : &H00 = File gevonden.
              &HFF = File niet gevonden.

&H13 DELETE FILE

Wist de file die in het File Control Block waar met het adres in registerpaar DE naartoe wordt verwezen uit de directory door in de directory het eerste teken van de file-naam te vervangen door de code &HE5 en in de FAT de door de file gebruikte clusters vrij te maken. CP/M-compatibel.

INPUT   DE: FCB-adres
OUTPUT   A : &H00 = File gewist.
              &HFF = File niet (geheel) gewist.

&H14 SEQUENTIAL READ

Leest een record (van 128 bytes). De file, het blok binnen de file waarin het record zich bevindt en het record zelf staan aangegeven in de velden File name, current block en current record van het File Control Block dat met het adres in register-

paar DE wordt aangewezen. De gelezen informatie wordt in het geheugen gezet vanaf het ingestelde "transfer"-adres. De velden current block en current record in het FCB worden automatisch met 1 verhoogd. CP/M-compatibel.

INPUT DE: FCB-adres.
OUTPUT A : &H00 = Record gelezen.
&H01 = Record niet (goed) gelezen.

&H15 SEQUENTIAL WRITE

Schrijft een record (van 128 bytes) naar de file die met het File Control Block dat met het adres in registerpaar DE wordt aangewezen. De data, die het record vormt, staat in het RAM-geheugen, vanaf het "transfer"-adres. Het record (op schijf) wordt bepaald door de velden current block en current record in het FCB. Deze velden worden automatisch met 1 verhoogd. CP/M-compatibel.

INPUT DE: FCB-adres.
OUTPUT A : &H00 = Record geschreven.
&H01 = Record niet geschreven.

&H16 CREATE FILE

Creeert een file, volgens de gegevens in het File Control Block, dat wordt aangewezen met het adres dat in registerpaar DE staat. Indien de file reeds bestaat, wordt deze overschreven. Na uitvoering van deze functie zijn de volgende velden vastgelegd:

Record Size, Current Block, Current Record, Random Record.

Of de file wel of niet is gecreeerd wordt teruggemeld in register A. CP/M-compatibel.

INPUT DE: FCB-adres
OUTPUT A : &H00 = File gecreeerd.
&HFF = File niet gecreeerd.

&H17 RENAME FILE

Geeft de file, die wordt aangegeven met het File Control Block dat wordt aangewezen met het adres in registerpaar DE, de file-naam uit het File Control Block dat wordt aangewezen door het adres in registerpaar DE plus 16. In de file-naam mag gebruik worden gemaakt van de tekens * en ?. CP/M-compatibel.

INPUT DE : FCB-adres te hernoemen file.

```
               DE+16: FCB-adres met nieuwe file-naam.
     OUTPUT    A    : &H00 = File hernoemd.
                      &HFF = File niet hernoemd.
```

&H18 GET LOGIN VECTOR

Geeft in registerpaar HL een bit-tabel waaruit kan worden opgemaakt welke drives on-line zijn. CP/M-compatibel.

```
OUTPUT    HL: Bit-tabel.
```

&H19 GET DEFAULT DRIVE NAME

Zet de default drive naam in register A. CP/M-compatibel.

```
OUTPUT    A : Default drive naam.
```

&H1A SET TRANSFER ADDRES

Stelt het RAM-adres, waar de data die naar disk moet of van disk wordt gelezen, in op het in registerpaar DE gegeven adres. CP/M-compatibel.

```
INPUT     DE: RAM-adres.
```

&H1B GET ALLOCATION

Geeft informatie over de in register E aangegeven disk drive. Indien de aangegeven disk naam niet geldig is, zal register A na uitvoering de waarde &HFF bevatten.

```
INPUT     E : Drive naam.
OUTPUT    A : Aantal sectors per cluster.
          BC: Aantal bytes per sector.
          DE: Aantal clusters op de schijf.
          HL: Aantal vrije clusters.
          IX: Device Parameter Block adres.
          IY: File Allocation Tabel adres.
```

&H21 RANDOM READ

Leest een record uit de file, die wordt aangegeven met het File Control Block dat wordt aangewezen met het adres in registerpaar DE. Het te lezen record wordt bepaald door het random block veld uit het FCB. De data van het record wordt in het RAM-geheugen gezet vanaf het "transfer"-adres. De record lengte is altijd 128 bytes. CP/M-compatibel.

```
INPUT     DE: FCB-adres.
OUTPUT    A : &H00 = Record gelezen.
              &H01 = Record niet (goed) gelezen.
```

&H22 RANDOM WRITE

Schrijft een record naar de file, die wordt aangegeven in het File Control Block, dat wordt aangewezen met het adres in registerpaar DE. De data voor het record begint op het "transfer"-adres. Welk record (binnen de file) wordt geschreven, wordt bepaald door het random block veld uit het FCB. CP/M-compatibel.

```
INPUT     DE: FCB-adres.
OUTPUT    A : &H00 = Record geschreven.
              &H01 = Record niet geschreven.
```

&H23 GET FILE SIZE

Berekent de lengte van de file uit het File Control Block dat wordt aangewezen door het adres in registerpaar DE. De lengte van de file wordt als een veelvoud van 128 gegeven en opgeslagen in het random record veld van het FCB. CP/M-compatibel.

```
INPUT     DE: FCB-adres.
OUTPUT    A : &H00 = Lengte berekend.
              &HFF = Lengte niet berekend.
```

&H24 SET RANDOM RECORD

Berekent de positie van het "current record" met behulp van de velden current block en current record uit het File Control Block dat wordt aangewezen met het adres in registerpaar DE en plaatst het resultaat van de berekening in het veld random record van hetzelfde FCB. CP/M-compatibel.

```
INPUT     DE: FCB-adres.
```

&H26 RANDOM BLOCK WRITE

Schrijft record(s) naar de file uit het File Control Block, dat wordt aangewezen door het adres in registerpaar DE. De data voor het record begint op het "transfer"-adres. Het te beschrijven record in de file wordt bepaald door het random block veld uit het FCB. Het veld current random record wordt automatisch verhoogd met het aantal geschreven records. Het aantal records wordt bepaald door de waarde in registerpaar HL. De recordlengte wordt bepaald door het record size veld uit het FCB.

```
INPUT     DE: FCB-adres.
          HL: Aantal records.
OUTPUT    A : &H00 = Record(s) geschreven.
              &H01 = Record(s) niet geschreven.
```

&H27 RANDOM RECORD READ

Leest record(s) uit de file die wordt aangegeven in het File Control Block, dat wordt aangewezen met het adres in registerpaar DE, in het geheugen naar het "transfer"-adres. Welk record (uit de file) wordt gelezen, wordt bepaald door de inhoud van het veld random block uit het FCB. Het veld current random record wordt automatisch verhoogd met het aantal gelezen records. De record lengte ligt vast in het record size veld. Het aantal records dat moet worden gelezen moet in HL staan. Het werkelijk aantal gelezen records wordt in registerpaar HL gezet.

```
INPUT      DE: FCB-adres.
           HL: Aantal te lezen records.
OUTPUT     A : &H00 = Record(s) gelezen.
               &H01 = Record(s) niet (goed) gelezen
           HL: Aantal gelezen records.
```

&H28 RANDOM WRITE WITH ZERO-FILL

Schrijft een record naar de file die wordt aangegeven in het File Control Block, dat wordt aangewezen met het adres in registerpaar DE. Welk record (in de file) wordt geschreven wordt bepaald door het veld random block in het FCB. De record lengte is altijd 128 bytes. Wanneer een file wordt uitgebreid (extend), zullen alle records die niet zijn geschreven met nullen worden gevuld. CP/M-compatibel.

```
INPUT      DE: FCB-adres.
OUTPUT     A : &H00 = Record geschreven.
               &H01 = Record niet geschreven.
```

&H2A GET DATE

Leest de datum en zet deze in Z80-registers.

```
OUTPUT     A : Dag van de week.
           D : Dag.
           E : Maand.
           HL: Jaar.
```

&H2B SET DATE

Stelt de datum in op de in de registerparen DE en HL gegeven datum.

```
INPUT      D : Dag
           E : Maand.
           HL: Jaar.
```

```
     OUTPUT     A : &H00 = Datum ingesteld.
                    &HFF = Datum niet ingesteld.

&H2C GET TIME
     Leest de tijd en zet deze in Z80-registers.
     OUTPUT     E : Honderdsten van seconden.
                D : Seconden.
                L : Minuten.
                H : Uren.

&H2D SET TIME
     Stelt  de tijd in op de in registerparen DE en  HL
     gegeven tijd.
     INPUT      E : Honderdsten van seconden.
                D : Seconden.
                L : Minuten.
                H : Uren.
     OUTPUT     A : &H00 = Tijd ingesteld.
                    &HFF = Tijd niet ingesteld.

&H2E SET VERIFY FLAG
     Zet  Verify  Flag aan of uit,  afhankelijk van  de
     inhoud van register E.
     INPUT      E : &H00 = Verify flag ge-reset.
                    ELSE = Verify flag ge-set.

&H2F ABSOLUTE DISK READ
     Leest  sector(s) van schijf en zet de  inhoud  van
     die   sector(s)  in  het  RAM-geheugen  vanaf  het
     "transfer"-adres.
     INPUT      DE: Sector nummer.
                H : Aantal te lezen sectoren.
                L : Drive nummer.

&H30 ABSOLUTE DISK WRITE
     Schrijft  sector(s) naar schijf,  vanuit het  RAM-
     geheugen (beginnende op het "transfer"-adres).
     INPUT      DE: Sector nummer.
                H : Aantal te schrijven sectoren.
                L : Drive nummer.
```

Hierna volgt nog een lijst van BDOS calls in alfabetische volgorde van naam.

Functie:	Nr.:
ABSOLUTE DISK READ	&H2F
ABSOLUTE DISK WRITE	&H30
AUX INPUT	&H03
AUX OUTPUT	&H04
BUFFERED INPUT	&H0A
CLOSE FILE	&H10
CONSOLE INPUT	&H01
CONSOLE OUTPUT	&H02
CONSOLE STATUS	&H0B
CREATE FILE	&H16
DELETE FILE	&H13
DIRECT CONSOLE IO	&H06
DIRECT INPUT	&H07
DIRECT INPUT	&H08
DISK RESET	&H0D
GET ALLOCATION	&H1B
GET DATE	&H2A
GET DEFAULT DRIVE NAME	&H19
GET FILE SIZE	&H23
GET LOGIN VECTOR	&H18
GET TIME	&H2C
GET VERSION NUMBER	&H0C
LST OUTPUT	&H05
OPEN FILE	&H0F
RANDOM BLOCK WRITE	&H26
RANDOM READ	&H21
RANDOM RECORD READ	&H27
RANDOM WRITE	&H22
RANDOM WRITE WITH ZERO-FILL	&H28
RENAME FILE	&H17
SEARCH FIRST	&H11
SEARCH NEXT	&H12
SELECT DISK	&H0E
SEQUENTIAL READ	&H14
SEQUENTIAL WRITE	&H15
SET DATE	&H2B
SET RANDOM RECORD	&H24
SET TIME	&H2D
SET TRANSFER ADDRESS	&H1A
SET VERIFY FLAG	&H2E
STRING OUTPUT	&H09
SYSTEM RESET	&H00

31 I/O-poort adressen

In de hiernavolgende opsomming van I/O-poort adressen en adres-gebieden, worden de MSX-specificaties gegeven. Het is echter heel goed mogelijk dat een bepaalde fabrikant afwijkt van deze specificatie. Het is daarom niet gegarandeerd, dat deze adressen exact overeenkomen met de in uw computer gebruikte adressen. Het is zelfs ten sterkste af te raden programma's voor gebruik op meerdere computers te schrijven, waarin deze adressen rechtstreeks worden aangeroepen. Dergelijke programma's dienen altijd gebruik te maken van de beschikbare BIOS-entry points.

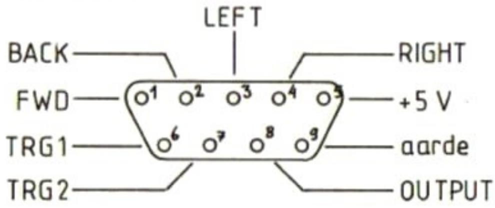
Adres(sen):	Functie:
&H00 - &H7F	Gereserveerd.
&H80 - &H83	RS232C-support (USART 8251):
&H80	DATA-register van de 8251-chip.
&H81	Status/commando-register van de 8251-chip.
&H82	Status/interrupt masker van de 8251-chip
&H83	Gereserveerd.
&H84 - &H87	Programmable Timer 8253.
&H88 - &H8B	Gereserveerd voor de Video Display Processor 9938 voor MSX1 adaptor.
&H8C - &H8D	Gereserveerd voor Modem.
&H8E - &H8F	Gereserveerd.
&H90 - &H91	Printer-poort:
&H90	Control: write - bit 0 = strobe. read - bit 1 = status.
&H91	Data.
&H92 - &H97	Gereserveerd.

Adres(sen):	Functie:
&H98 - &H9B	Gereserveerd voor de Video Display Processor. (MSX1 en MSX2).
&H9C - &H9F	Gereserveerd.
&HA0 - &HA3	Sound processor (AY-3-8910):
&HA0	Adres-register
&HA1	Data naar register schrijven.
&HA2	Data uit register lezen.
&HA4 - &HA7	Gereserveerd.
&HA8 - &HAB	Parallel I/O-poort (8255):
&HA8	Poort A lezen en schrijven.
&HA9	Poort B lezen en schrijven.
&HAA	Poort C lezen en schrijven.
&HAB	Mode setting register.
&HAC - &HAF	Gereserveerd.
&HB0 - &HB3	Extern geheugen (8255):
&HB0	Poort A: Adreslijnen A0-A7.
&HB1	Poort B: Adreslijnen A8-A10, A13-A15, CONTROL, R/W.
&HB2	Poort C: Adreslijnen A11-A12, datalijnen D0-D7
&HB3	Mode setting register.
&HB4 - &HB5	Kalender klok (RP-5C01):
&HB4	Adres register.
&HB5	Data lezen en schrijven.
&HB6 - &HB7	Gereserveerd.
&HB8 - &HBB	Licht-pen.
&HBC - &HBF	VHD Control.
&HC0 - &HC1	MSX-audio.
&HC2 - &HC7	Gereserveerd.
&HC8 - &HCF	MSX-interface.

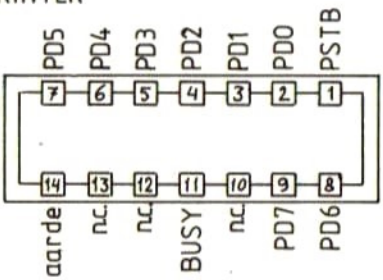
Adres(sen):	Functie:
&HD0 - &HD7	Floppy disk controller.
&HD8 - &HDB	KANJI-ROM:
&HD8	Adreslijnen A5-A0 (minst significant)
&HD9	Adreslijnen A5-A0 (meest significant) of bij lezen datalijnen D7-D0.
&HDC - &HF4	Gereserveerd.
&HF5	Systeem Controle: De bits van deze I/O-poort geven aan welke devices in het systeem aanwezig zijn. Een 1 wil zeggen aanwezig, een 0 betekent niet aanwezig. Deze poort kan zowel worden gelezen als beschreven. Door het betreffende bit 0 te maken verwijdert u het device als het ware uit het systeem: bit 0 - KANJI-ROM. bit 1 - gereserveerd. bit 2 - MSX-AUDIO. bit 3 - Super impose. bit 4 - MSX-interface. bit 5 - RS232C-interface. bit 6 - Licht-pen bit 7 - Kalender-klok.
&HF6	Color bus I/O
&HF7	A/V-control. De betekenis van de bits op dit I/O-adres is als volgt: bit 0 - Audio Rechts bit 1 - Audio Links bit 2 - Video Input Select bit 3 - Video Input Sense (lezen) bit 4 - A/V-control. bit 5 - Ym control. bit 6 - Ys control. bit 7 - Video Select.
&HF8 - &HFB	Gereserveerd.
&HFC - &HFF	Memory mapper.

32 Connectoren

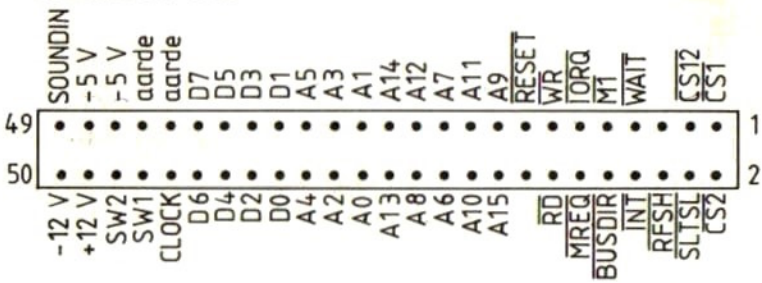
JOY STICK



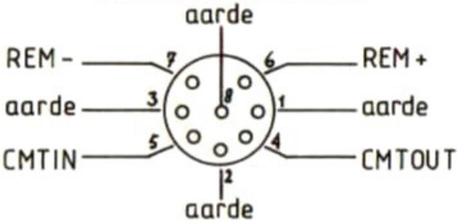
PRINTER



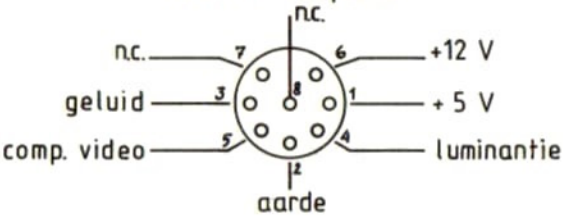
EXPANSION BUS



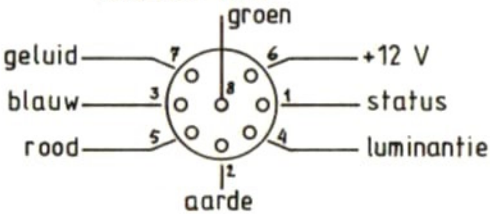
CASSETTE-RECORDER



MONITOR composite

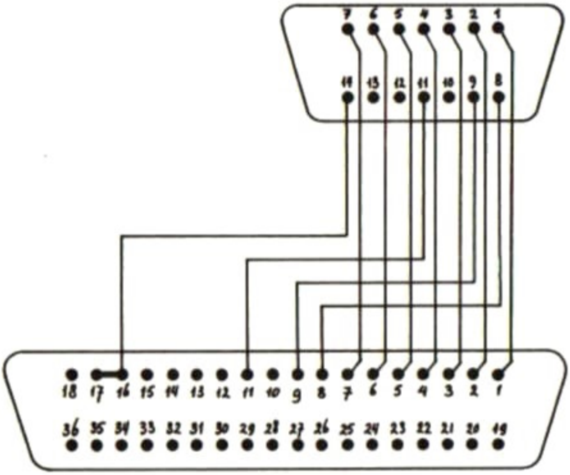


MONITOR RGB



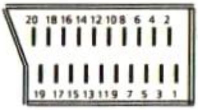
Het volgende schema laat zien hoe een parallelprinter op de MSX-computer moet worden aangesloten. De meeste parallelprinters hebben een 36-polige amphenol-plug. De MSX-computer heeft een 14-polige amphenol-plug. Het beste is het om een kabel met "twisted pair" draden te gebruiken. Aan de printer-kant worden dan de twee in elkaar gedraaide draden op twee tegenoverelkaar liggende aansluitingen gesoldeerd (bijv. pin 1 en 19) Aan de computerkant worden dan de draden die aan pinnen 19 en hoger zitten afgeknipt, terwijl de overige draden als in het schema worden verbonden.

MSX-computer



Printer (Centronics-poort)

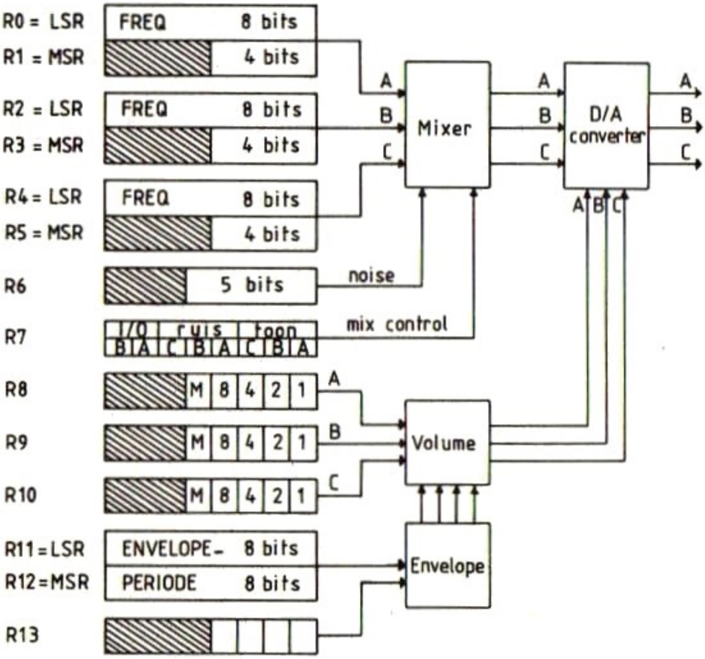
SCART-connector



1	Audio uit (rechts)	8	Status CVBS	15	Rood
2	Audio in (rechts)	9	Groen (aarde)	16	Status RGB
3	Audio uit (links)	10	-	17	CVBS (aarde)
4	Audio (aarde)	11	Groen	18	Status RGB (aarde)
5	Blauw (aarde)	12	-	19	CVBS uit
6	Audio in (links)	13	Rood (aarde)	20	CVBS in
7	Blauw	14	Aarde		

33 De programmeerbare geluidsgenerator

In plaats van het PLAY-statement, kan men ook gebruik maken van het SOUND-statement. Hiermee zijn meer en gevarieerdere geluidseffecten te verkrijgen. Om een goed gebruik van het SOUND-statement te kunnen maken dient men echter iets meer van de geluidsgenerator te weten. De hierna volgende afbeelding laat, als het ware, de opbouw van de generator zien.



Aan de linkerkant in de afbeelding van de geluidsgenerator ziet u de te programmeren registers. Deze registers zijn genummerd van R0 tot en met R15. R14 en R15 hebben geen geluidsfunctie. Zij worden gebruikt voor de joystick-interface. Zie hiervoor het hoofdstukje over de I/O-adressen.

In de afbeelding en de hiernavolgende uitleg zult u de afkortingen RP, MSR en LSR tegenkomen. Deze afkortingen staan voor Register Paar, Meest Significant Register en Minst (Least) Significant Register.

De registers kunnen worden geladen door middel van een SOUND-statement. Register 7 dient als laatste te worden geladen. Met register 7 kunnen de verschillende geluids kanalen aan of uit worden gezet. Daarbij geldt dat, indien een van de bitjes een nul is, de betreffende functie is aangeschakeld, terwijl een bitje dat de waarde 1 krijgt de betreffen de functie uitschakelt. Hier volgt een korte verklaring van register 7.

I/O		$\overline{\text{NOISE}}$			$\overline{\text{TOON}}$		
A	B	C	B	A	C	B	A

```
128 64 32 16 8   4  2  1 <-- decimale waarde van het bit
 |   |  |  |  |  |  |  |
 |   |  |  |  |  |  |  |---- 0 = geluidskanaal A aan.
 |   |  |  |  |  |  |        1 = geluidskanaal A uit.
 |   |  |  |  |  |  |------- 0 = geluidskanaal B aan.
 |   |  |  |  |  |           1 = geluidskanaal B uit.
 |   |  |  |  |  |---------- 0 = geluidskanaal C aan.
 |   |  |  |  |              1 = geluidskanaal C uit.
 |   |  |  |  |------------- 0 = ruis op kanaal A.
 |   |  |  |                 1 = geen ruis op kanaal A.
 |   |  |  |---------------- 0 = ruis op kanaal B.
 |   |  |                    1 = geen ruis op kanaal B.
 |   |  |--------------------0 = ruis op kanaal C.
 |   |                       1 = geen ruis op kanaal C.
 |   |---------------------- 0 = I/O-poort A niet actief
 |                           1 = I/O-poort A actief.
 |-------------------------- 0 = I/O-poort B niet actief
                             1 = I/O-poort B actief.
```

Om dus een toon op kanaal A te laten klinken en tegelijkertijd een ruis op kanaal B te laten klinken, zullen de bitjes 0 en 4 laag dienen te zijn. Alle andere bits mogen hoog zijn. dit zou een decimale

waarde geven van 128+64+32+8+4+2=238. Met SOUND 7,238 zouden de gewenste instellingen worden bereikt.

Het instellen van het volume van de verschillende kanalen kan worden gedaan door de registers R8, R9 en R10 met een waarde van 0 tot en met 15 te laden. Hoe hoger de waarde, hoe sterker het volume. Door de waarde 16 in te geven, wordt het bitje dat is aangeduid met M hoog gemaakt. Hieruit volgt dat het volume voor het betreffende kanaal op Maximaal staat.

Het instellen van de frequentie van de toon kan worden bereikt door de registers R0 tot en met R5 met een bepaalde waarde te laden. R0 en R1 zijn voor kanaal A, R2 en R3 voor kanaal B en R4 en R5 voor kanaal C. Om een kanaal een gewenste frequentie te laten produceren, dient men een waarde voor de betreffende registers te berekenen. De berekende waarde moet dan worden verdeeld over de beide registers. Hiervoor kunt u de volgende methode gebruiken.

RP = INT (111860/<frequentie>)

MSR = INT (RP/256)

LSR = RP - 256 * MSR

LSR kunt u gebruiken voor de registers R0, R2 of R4. MSR laadt u dan in de bijbehorende registers R1, R3 of R5. Een voorbeeldje moge een en ander duidelijk maken. Stel we willen register B een toon van 1000 Herz laten maken. De berekening gaat dan als volgt:

RP = INT (111860/1000) = 111

MSR = INT (111/256) = 0

LSR = RP - 256 * MSR = 111 - 0 = 111

Nu geeft u twee SOUND statements, SOUND 2,111
SOUND 3,0

Door hierna kanaal B te activeren (R7), zal er een geluid van 1000 Hz worden geproduceerd.

Het zal u opgevallen zijn dat deze berekening zich uitstekend leent voor automatiseren. Met het volgende

BASIC-programma kunt u iedere gewenste frequentie laten omzetten naar registerinhouden voor de registers LSR en MSR. Het programma vraagt eerst naar de beginfrequentie, vervolgens naar de eindfrequentie en tenslotte naar de stapgrootte. Daarna krijgt u een tabel met registerwaarden per frequentie.

HOOFDDEEL

```
10 '*******************************
20 '*Berekenen inhoud van de toon-*
30 '*registers 0 t/m 15 van de PSG*
40 '*******************************
50 '
60 CLS
70 INPUT "DE LAAGSTE FREQUENTIE";L
80 INPUT "DE HOOGSTE FREQUENTIE";H
90 INPUT "DE STAPGROOTTE IN HZ.";S
100 PRINT
110 FOR I=L TO H STEP S
120 P=INT(1789770#/(16*I))
130 R1=INT(P/256)
140 R0=P-R1*256
```

plus

TABEL

```
150 PRINT "F=";USING"####";I;
160 PRINT "  R0=";USING "###";R0;
170 PRINT "  R1=";USING "##";R1
180 NEXT I
190 END
```

of

GELUID

```
150 SOUND 0,R0
160 SOUND 1,R1
170 SOUND 7,62
180 SOUND 8,15
190 NEXT I
200 END
```

Indien u ruis wenst, kunt u in register 6 de frequentie van die ruis opgeven. Ook daarvoor zou weer een berekening te maken zijn. Er zijn echter maar 31 mogelijkheden. De volgende tabel geeft de mogelijke register inhoud van R6 weer, met daarachter de bijbehorende frequentie.

Inhoud R6	Ruisfrequentie
1	111860 Hz
2	55930 Hz
3	37286 Hz
4	27965 Hz
5	22372 Hz
6	18643 Hz
7	15980 Hz
8	13982 Hz
9	12428 Hz
10	11186 Hz
11	10169 Hz
12	9321 Hz
13	8604 Hz
14	7990 Hz
15	7457 Hz
16	6991 Hz
17	6580 Hz
18	6214 Hz
19	5887 Hz
20	5593 Hz
21	5326 Hz
22	5084 Hz
23	4863 Hz
24	4660 Hz
25	4474 Hz
26	4302 Hz
27	4142 Hz
28	3995 Hz
29	3857 Hz
30	3728 Hz
31	3608 Hz

De inhoud van de registers waarmee de envelop-generator wordt gestuurd kan als volgt worden berekend:

RP = <tijd in seconden> * 6991

R12 = INT (RP/256)

R11 = RP - 256 * R12

Ook hier weer een voorbeeldje ter verduidelijking. Stel we wensen een envelop-tijd van 4 seconden. We krijgen:

RP = 4 * 6991 = 27964

R12 = INT (27964/256) = 109

R11 = 27964 - 109 * 256 = 27964 - 27904 = 60

Het zetten van de envelop-tijd wordt dan met twee SOUND statements als volgt gedaan, SOUND 11,60
SOUND 12,109

De volgende tabel zal waarschijnlijk de meest gebruikte waarden al bevatten, zodat u niet vaak meer zult hoeven te rekenen.

Tijd in seconden	Inhoud van R11	en R12
9	199	245
8	120	218
7	41	191
6	218	163
5	139	136
4	60	109
3	237	81
2	158	54
1	79	27
0.9	147	24
0.8	216	21
0.7	29	19
0.6	98	16
0.5	167	13
0.4	236	10
0.3	49	8
0.2	118	5
0.1	187	2

34 Video Display Processor

In MSX-computers worden twee verschillende Video Display Processors gebruikt. De TMS9918-chip wordt als video processor in MSX1 computers toegepast. In MSX2 computers wordt de TMS9938-chip gebruikt. Alles wat de TMS9918 kan, kan de TMS9938 ook, maar deze laatste chip kan nog meer.

De TMS9918 kent vier screen-modes, waarbij de scherminformatie wordt opgeborgen in een video-geheugen dat maximaal 16 kbytes groot is.

De TMS9938 kent 9 screen-modes, de vier modes van de TMS9918 plus vijf nieuwe modes. Die nieuwe modes zijn bijna allemaal zogenaamde bit-mapped modes. Dat houdt in, dat ieder beeldpunt afzonderlijk is te benaderen en afzonderlijk een kleur kan worden toegekend. Afhankelijk van de gekozen mode zal er tot 64 kbytes aan videogeheugen nodig zijn voor het opslaan van 1 beeldschermplaatje. De TMS9938 is in staat beeldschermplaatjes in verschillende pagina's op te slaan. Het videogeheugen kan dan ook tot maximaal 128 kbytes groot zijn.

In de hiernavolgende beschrijving zal eerst worden gekeken naar de indeling van het Video RAM voor de verschillende screen-modes. Daarna zal de video display processor zelf onder de loep worden genomen. Dat laatste wil zeggen, dat de besturing van de chip zal worden bekeken aan de hand van de interne registers daarvan.

Indeling van het Video RAM.

In het video RAM wordt de informatie, die op het scherm moet worden afgebeeld, in de vorm van tabellen opgeslagen. Zo zijn er tabellen, waarin de matrix-patronen van de letters worden opgeslagen, tabellen waarin de kleuren van letters of beeldpuntjes staan en tabellen waarin de sprites zijn vastgelegd.

In de navolgende beschrijvingen zullen deze tabellen verder worden uitgewerkt. De startadressen van de tabellen zijn onder BASIC door het systeem vastgelegd.

Wilt u het startadres weten, dan kunt u dat opvragen met de systeemvariabele BASE(n). Hierin staat de letter n voor het nummer van de gevraagde tabel. Deze nummers zullen in de volgende beschrijving worden gegeven.

Tenslotte dient nog te worden opgemerkt, dat de startadressen van alle tabellen met behulp van de systeemvariabele BASE kunnen worden uitgelezen. Alleen de adressen van de tabellen 0 tot en met 19 kunnen worden gewijzigd (beschreven). Indien de adressen van tabellen 10 tot en met 14 worden gewijzigd, heeft dat tot gevolg, dat ook de adressen van de tabellen 20 tot en met 24 worden gewijzigd.

SCREEN 0 (40-tekens)

Tabelnaam:	default-adres	lengte:	reg.nr.
Scherm-info-tabel	&H0000-&H03BF	960 bytes	BASE(0)
Matrix-tabel	&H0800-&H0FFF	2048 bytes	BASE(2)
Palette-tabel	&H0400-&H041F	32 bytes	-

Scherm-info-tabel:
Deze tabel bestaat uit entries van 1 byte. Iedere entry heeft betrekking op een tekenpositie op het beeldscherm. Er zijn 960 tekenposities op het scherm, en daarom ook 960 entries in deze tabel. De eerste entry is voor het teken dat linksbovenaan het scherm staat, de tweede entry voor het teken daarnaast op dezelfde regel, etc.
De inhoud van een entry geldt als een pointer naar een entry in de matrix-tabel. Zou een entry in de scherm-info-tabel de waarde 100 hebben, dan wijst dit dus naar de entry nummer 100 in de matrix-tabel.

Matrix-tabel:
Deze tabel bestaat uit 256 entries van elk 8 bytes. Iedere entry bevat de 8 bij 8 matrix van een teken, zoals dat op het beeldscherm wordt afgedrukt. Elk byte bevat 8 horizontale beeldpunten. Elk volgende byte bevat de 8 horizontale punten van het teken op de volgende beeldlijn.

Palette-tabel:
Deze tabel is alleen in MSX2-computers aanwezig. Deze tabel bestaat uit 16 entries, voor ieder van de kleuren

0 tot en met 15. Elke entry is twee bytes lang. In het eerste byte staat de intensiteit van de kleuren rood en blauw. Rood staat in het linker tetrade. In het tweede byte staat de intensiteit van de kleur groen. De intensiteit van iedere kleur kan worden aangegeven met een waarde van 0 tot 7. Om rood en groen samen in een byte te kunnen coderen, dient de waarde voor rood met 16 te worden vermenigvuldigd, waarna er de waarde voor groen wordt bijgeteld.

SCREEN 0 (80 tekens)

Tabelnaam:	default-adres	lengte:	reg.nr.
Scherm-info-tabel	&H0000-&H077F	1920 bytes	BASE(0)
Kleur-tabel	&H0800-&H090D	270 bytes	BASE(1)
Matrix-tabel	&H1000-&H17FF	2048 bytes	BASE(2)
Palette-tabel	&H0F00-&H0F1F	32 bytes	-

Scherm-info-tabel:
Onder MSX2 kan met WIDTH=80 de regellengte tot 80 tekens worden verlengd. In dat geval zal de scherm-info-tabel 1920 bytes lang zijn (twee keer zo lang dus). Voor het overige is deze tabel gelijk van opbouw als beschreven onder SCREEN 0 (40 tekens).

Kleur-tabel:
Deze tabel is 240 bytes lang, hetgeen overeenkomt met 240 keer 8 is 1920 bits. Dit is 1 bit voor iedere tekenpositie op het scherm. Een bit in deze tabel geeft aan of het bijbehorende teken op het scherm de normaal ingestelde kleur moet hebben (bit=0), of de kleur zoals aangegeven in VDP-register 12. VDP-register 13 geeft in dat geval aan hoe lang de normaal ingestelde kleur en de kleur uit het VDP-register afwisselend moeten worden weergegeven.

Matrix-tabel:
Zie onder SCREEN 0 (40 tekens).

Palette-tabel:
Zie onder SCREEN 0 (40 tekens).

SCREEN 1

Tabelnaam:	default-adres	lengte:	reg.nr.
Matrix-tabel	&H0000-&H07FF	2048 bytes	BASE(7)
Scherm-info-tabel	&H1800-&H1AFF	768 bytes	BASE(5)
Sprite-info-tabel	&H1B00-&H1B7F	128 bytes	BASE(8)
Kleur-tabel	&H2000-&H201F	32 bytes	BASE(6)
Palette-tabel	&H2020-&H203F	32 bytes	-
Sprite$-tabel	&H3800-&H3FFF	2048 bytes	BASE(9)

Matrix-tabel:
Zie onder SCREEN 0 (40 tekens).

Scherm-info-tabel:
Onder deze screen-mode heeft iedere regel maximaal 32 tekens. Vandaar dat in deze tabel slechts 32 keer 24 is 768 entries van 1 byte staan. De functie en betekenis van iedere entry is dezelfde als beschreven onder SCREEN 0 (40 tekens).

Sprite-info-tabel:
Deze tabel bestaat uit 32 entries van elk 4 bytes. Het eerst byte bevat de Y-coordinaat van het sprite. Het tweede byte de X-coordinaat. Het derde byte bevat het sprite-nummer. Het vierde byte bevat de volgende informatie:

- in de bits 0 tot en met 3 het kleurnummer.
- bit 5: Indien dit bit 1 is, worden sprite-botsingen niet meer gedetecteerd.
- bit 6: Indien dit bit 1 is, dan wordt het sprite niet weergegeven. Passeert dit sprite een andere sprite, waarvan dit bit ook op 1 staat, dan wordt een botsing niet geconstateerd. Passeert dit sprite een andere sprite waarvan dit bit 0 is, dan zijn er twee mogelijkheden:
  1. Passeert langs de achterkant:
     Het sprite wordt zichtbaar over de samenvallende beeldlijnen.
  2. Passeert langs de voorkant:
     Het sprite blijft onzichtbaar en een botsing wordt niet gedetecteerd.
- bit 7: Indien dit bit 1 is, wordt het sprite 32 posities verder naar links geprojecteerd. Links van het beeld geschoven sprite-delen worden niet aan de rechterkant van het scherm weergegeven.

Kleur-tabel:
Zie onder SCREEN 0 (80 tekens).

Palette-tabel:
Zie onder SCREEN 0 (40 tekens).

Sprite$-tabel:
Deze tabel bevat de matrices van de gedefinieerde sprites. Bij 8*8-sprites kunnen er 256 sprite-definities in, bij 16*16 sprites passen er 64 sprite-definities in.

SCREEN 2

Tabelnaam:	default-adres	lengte:	reg.nr.
Matrix-tabel	&H0000-&H17FF	6144 bytes	BASE(12)
Scherm-info-tabel	&H1800-&H1AFF	768 bytes	BASE(10)
Sprite-info-tabel	&H1B00-&H1B7F	128 bytes	BASE(13)
Palette-tabel	&H1B80-&H1B9F	32 bytes	-
Kleur-tabel	&H2000-&H37FF	6144 bytes	BASE(11)
Sprite$-tabel	&H3800-&H3FFF	2048 bytes	BASE(14)

Matrix-tabel:
In schermmode 2 is het beeldscherm opgebouwd uit 24 regels van 32 tekens. Ieder teken bestaat uit een matrix van 8 bij 8 bits. De matrix-tabel is opgebouwd uit 24*32=768 entries van 8 bytes (de teken-matrix). De bitjes in de bytes komen overeen met de beeldpuntjes op het scherm, binnen de teken-matrix.

Scherm-info-tabel:
Deze tabel bestaat uit drie blokken van 256 bytes, die door BASIC met de default waarden 0 tot en met 255 worden gevuld. De waarden 0 tot en met 255 uit het eerste blok hebben betrekking op de eerste 256 entries in de matrix-tabel. De tweede serie waarden van 0 tot en met 256 hebben betrekking op de tweede 256 entries in de matrix-tabel, etc. De plaats van een waarde in de scherm-info-tabel bepaalt waar de uit de matrix-tabel gelezen entry op het scherm zal worden afgedrukt.

Sprite-info-tabel:
Zie onder SCREEN 1.

Palette-tabel:
Zie onder SCREEN 0 (40 tekens).

Kleur-tabel:
Deze tabel loopt parallel aan de matrix-tabel. In ieder byte staan twee kleuren, de voorgrond- en de achtergrondkleur. Dit houdt dus in dat elk horizontaal rijtje van 8 bits uit de matrix-tabel slechts 2 verschillende kleuren kan hebben. De waarde van ieder byte in de kleur-tabel is 16 keer de voorgrondkleur plus de achtergrondkleur.

Sprite$-tabel:
Zie onder SCREEN 1.

SCREEN 3.

Tabelnaam:	default-adres	lengte:	reg.nr.
Matrix-tabel	&H0000-&H07FF	2048 bytes	BASE(17)
Scherm-info-tabel	&H0800-&H0AFF	768 bytes	BASE(15)
Sprite-info-tabel	&H1B00-&H1B7F	128 bytes	BASE(18)
Palette-tabel	&H2020-&H203F	32 bytes	-
Sprite$-tabel	&H3800-&H3FFF	2048 bytes	BASE(19)

Matrix-tabel:
Per af te drukken teken bevat deze tabel een entry van twee bytes.
De linker tetrade van het eerste byte bevat de kleur voor het vakje van 4*4 beeldpuntjes aan de linker bovenkant van de tekenpositie.
De rechter tetrade van het eerste byte bevat de kleur voor het vakje van van 4*4 beeldpuntjes aan de rechter bovenkant van de tekenpositie.
De linker tetrade van het tweede byte bevat de kleur voor het vakje van 4*4 beeldpuntjes aan de linker onderkant van de tekenpositie.
De rechter tetrade van het tweede byte bevat de kleur voor het vakje van 4*4 beeldpuntjes aan de rechter onderkant van de tekenpositie.

Scherm-info-tabel:
Voor iedere tekenpositie op het beeldscherm (regel voor regel van links naar rechts) bevat deze tabel een entry van 1 byte. Een beeldschermpositie bestaat uit 4 vierkantjes van elk vier bij vier beeldpuntjes. Iedere

entry uit deze tabel verwijst naar een twee-bytes entry in de matrix-tabel.

Sprite-info-tabel:
Zie onder SCREEN 1.

Palette-tabel:
Zie onder SCREEN 0 (40 tekens).

Sprite$-tabel:
Zie onder SCREEN 1.

SCREEN 4 (256*192 pixels).

Tabelnaam:	default-adres	lengte:	reg.nr.
Matrix-tabel	&H0000-&H17FF	6144 bytes	BASE(22)
Scherm-info-tabel	&H1800-&H1AFF	768 bytes	BASE(20)
Sprite-kleur-tabel	&H1C00-&H1DFF	512 bytes	-
Sprite-info-tabel	&H1E00-&H1E7F	128 bytes	BASE(23)
Palette-tabel	&H1E80-&H1E9F	32 bytes	-
Kleur-tabel	&H2000-&H37FF	6144 bytes	BASE(21)
Sprite$-tabel	&H3800-&H3FFF	2048 bytes	BASE(24)

Matrix-tabel:
Zie onder SCREEN 2.

Scherm-info-tabel:
Zie onder SCREEN 2.

Sprite-kleur-tabel:
Deze tabel bevat een entry van 16 bytes voor ieder op het scherm afgedrukte sprite. Sprites worden op een transparant afgedrukt en er zijn in totaal 32 transparanten. Een 8*8 sprite bestaat uit 8 horizontale beeldlijnen. Een 16*16 sprite bestaat uit 16 horizontale beeldlijnen. Een 8*8 sprite gebruikt dus maar 8 bytes van een entry in deze tabel. Voor iedere beeldlijn van een sprite geeft het overeenkomstige byte in deze tabel onder meer de kleur van de lijn. De bytes in deze tabel hebben de volgende indeling:

- in de bits 0 tot en met 3 het kleurnummer.
- bit 5: Indien dit bit 1 is, worden sprite-botsingen niet meer gedetecteerd.
- bit 6: Indien dit bit 1 is, dan wordt het sprite niet weergegeven. Passeert de sprite een an-

dere sprite, waarvan dit bit ook op 1 staat, dan wordt een botsing niet geconstateerd. Passeert dit sprite een andere sprite waarvan dit bit 0 is, dan zijn er twee mogelijkheden:

1. Passeert langs de achterkant:
   Het sprite wordt zichtbaar over de samenvallende beeldlijnen.
2. Passeert langs de voorkant:
   Het sprite blijft onzichtbaar en een botsing wordt niet gedetecteerd.

- bit 7: Indien dit bit 1 is, wordt het sprite 32 posities verder naar links geprojecteerd. Links van het beeld geschoven sprite-delen worden niet aan de rechterkant van het scherm weergegeven.

Sprite-info-tabel:
Voor ieder transparant staat in deze tabel een entry van vier bytes. Er zijn 32 transparanten, zodat de tabel 32*4=128 bytes lang is. De betekenis van iedere entry is als volgt:
Eerste byte - Y-coordinaat van de sprite.
Tweede byte - X-coordinaat van de sprite.
Derde byte - Het sprite-nummer van de afgebeelde sprite.
Vierde byte - Wordt niet gebruikt.

Palette-tabel:
Zie onder SCREEN 0 (40-tekens).

Kleur-tabel:
Zie onder SCREEN 2.

Sprite$-tabel:
Zie onder SCREEN 1.

SCREEN 5 tot en met 8 zijn zogenaamde bit-mapped schermmodes. Een andere byzonderheid is, dat in deze modes meer dan 1 beeldschermplaatje tegelijk in het video geheugen aanwezig kan zijn (ook onder BASIC). Er staan als het ware meerdere pagina's in het geheugen, waarvan er 1 actief is, ofwel op het beeldscherm wordt weergegeven.

SCREEN 5 (256*212 pixels)

Indien de MSX2-computer 64 kbytes videogeheugen heeft, kunnen er twee pagina's in op worden geslagen. Indien het videogeheugen 128 kbytes groot is, kunner er 4 pagina's in worden opgeslagen. De startadressen van de pagina's is als volgt:

pagina 1 - &H0000
pagina 2 - &H8000
pagina 3 - &H10000
pagina 4 - &H18000

Iedere pagina heeft de volgende lay-out:

Tabelnaam:	default-adres	lengte:	reg.nr.
Scherm-info-tabel	&H0000-&H69FF	27136 bytes	BASE(25)
Sprite-kleur-tabel	&H7400-&H75FF	512 bytes	-
Sprite-info-tabel	&H7600-&H767F	128 bytes	BASE(28)
Palette-tabel	&H7680-&H769F	32 bytes	-
Sprite$-tabel	&H7800-&H7FFF	2048 bytes	BASE(29)

Scherm-info-tabel:
Voor ieder pixel op het beeldscherm staat in deze tabel een groepje van 4 bits. De volgorde van de pixels wordt gerekend van linksboven, beeldlijn voor beeldlijn (212 lijnen) naar rechtsonder. Elk groepje van 4 bits in deze tabel bevat het kleurnummer van het overeenkomstige pixel op het scherm. De kleurnummers gaan van 0 tot en met 15.
Op een beeldlijn staan 256 pixels. Voor iedere beeldlijn zijn zodoende 128 bytes nodig. Er zijn 212 beeldlijnen. Dit maakt dat de lengte van de tabel 128*212=27136 bytes lang is. In ieder byte staan de kleurnummers van een even en een oneven genummerd pixel. De totale waarde van een byte wordt bepaald door 16 keer het kleurnummer van het even pixel plus het kleurnummer van het oneven pixel.

Sprite-kleur-tabel:
Zie onder SCREEN 4.

Sprite-info-tabel:
Zie onder SCREEN 4.

Palette-tabel:
Zie onder SCREEN 0 (40-tekens).

Sprite$-tabel:
Zie onder SCREEN 1.

SCREEN 6 (512*212 pixels).

Indien de MSX2-computer 64 kbytes videogeheugen heeft, kunnen er twee pagina's in op worden geslagen. Indien het videogeheugen 128 kbytes groot is, kunner er 4 pagina's in worden opgeslagen. De startadressen van de pagina's is als volgt:

pagina 1 - &H0000
pagina 2 - &H8000
pagina 3 - &H10000
pagina 4 - &H18000

Iedere pagina heeft de volgende lay-out:

Tabelnaam:	default-adres	lengte:	reg.nr.
Scherm-info-tabel	&H0000-&H69FF	27136 bytes	BASE(30)
Sprite-kleur-tabel	&H7400-&H75FF	512 bytes	-
Sprite-info-tabel	&H7600-&H767F	128 bytes	BASE(33)
Palette-tabel	&H7680-&H769F	32 bytes	-
Sprite$-tabel	&H7800-&H7FFF	2048 bytes	BASE(34)

Scherm-info-tabel:
Voor ieder pixel op het beeldscherm staat in deze tabel een groepje van 2 bits. De volgorde van de pixels wordt gerekend van linksboven, beeldlijn voor beeldlijn (212 lijnen) naar rechtsonder. Elk groepje van 2 bits in deze tabel bevat het kleurnummer van het overeenkomstige pixel op het scherm. De kleurnummers gaan van 0 tot en met 3.
Op een beeldlijn staan 512 pixels. Voor iedere beeldlijn zijn zodoende 128 bytes nodig. Er zijn 212 beeldlijnen. Dit maakt dat de lengte van de tabel 128*212=27136 bytes lang is. In ieder byte staan de kleurnummers van vier pixels (1, 2, 3 en 4). De totale waarde van een byte wordt bepaald door 64 keer het kleurnummer van pixel 1 plus 16 keer het kleurnummer van pixel 2 plus 4 keer het kleurnummer van pixel 3 plus het kleurnummer van pixel 4.

Sprite-kleur-tabel:
Zie onder SCREEN 4.

Sprite-info-tabel:
Zie onder SCREEN 4.

Palette-tabel:
Zie onder SCREEN 0 (40-tekens).

Sprite$-tabel:
Zie onder SCREEN 1.

SCREEN 7 en 8 zijn alleen mogelijk op MSX2-computers met 128 kbytes videogeheugen.

SCREEN 7 (512*212 pixels)

Er kunnen twee pagina's in het video geheugen worden opgeslagen, die de volgende startadressen hebben:

Pagina 1 - &H0000
Pagina 2 - &H10000

Iedere pagina heeft de volgende lay-out:

Tabelnaam:	default-adres	lengte:	reg.nr.
Scherm-info-tabel	&H0000-&HD3FF	54272 bytes	BASE(35)
Sprite$-tabel	&HF000-&HF7FF	2048 bytes	BASE(39)
Sprite-kleur-tabel	&HF800-&HF9FF	512 bytes	-
Sprite-info-tabel	&HFA00-&HFA7F	128 bytes	BASE(38)
Palette-tabel	&HFA80-&HFA9F	32 bytes	-

Scherm-info-tabel:
Voor ieder pixel op het beeldscherm staat in deze tabel een groepje van 4 bits. De volgorde van de pixels wordt gerekend van linksboven, beeldlijn voor beeldlijn (212 lijnen) naar rechtsonder. Elk groepje van 4 bits in deze tabel bevat het kleurnummer van het overeenkomstige pixel op het scherm. De kleurnummers gaan van 0 tot en met 15.
Op een beeldlijn staan 512 pixels. Voor iedere beeldlijn zijn zodoende 256 bytes nodig. Er zijn 212 beeldlijnen. Dit maakt dat de lengte van de tabel 256*212=54272 bytes lang is. In ieder byte staan de

kleurnummers van een even en een oneven pixel. De totale waarde van een byte wordt bepaald door 16 keer het kleurnummer van het even pixel plus het kleurnummer van het oneven pixel.

Sprite$-tabel:
Zie onder SCREEN 1.

Sprite-kleur-tabel:
Zie onder SCREEN 4.

Sprite-info-tabel:
Zie onder SCREEN 4.

Palette-tabel:
Zie onder SCREEN 0 (40-tekens).

SCREEN 8 (256*212 pixels)

Er kunnen twee pagina's in het video geheugen worden opgeslagen, die de volgende startadressen hebben:

Pagina 1 - &H0000
Pagina 2 - &H10000

Iedere pagina heeft de volgende lay-out:

Tabelnaam:	default-adres	lengte:	reg.nr.
Scherm-info-tabel	&H0000-&HD3FF	54272 bytes	BASE(40)
Sprite$-tabel	&HF000-&HF7FF	2048 bytes	BASE(44)
Sprite-kleur-tabel	&HF800-&HF9FF	512 bytes	-
Sprite-info-tabel	&HFA00-&HFA7F	128 bytes	BASE(43)
Palette-tabel	&HFA80-&HFA9F	32 bytes	-

Scherm-info-tabel:
Voor ieder pixel op het beeldscherm staat in deze tabel een byte. De volgorde van de pixels wordt gerekend van linksboven, beeldlijn voor beeldlijn (212 lijnen) naar rechtsonder. Elk byte in deze tabel bevat een kleurcode voor het overeenkomstige pixel op het scherm. De kleurcode gaat van 0 tot en met 255.
Op een beeldlijn staan 256 pixels. Voor iedere beeldlijn zijn zodoende 256 bytes nodig. Er zijn 212 beeldlijnen. Dit maakt dat de lengte van de tabel 256*212=54272 bytes is.

De kleurcode wordt samengesteld uit 32 keer het aantal delen groen plus 4 keer het aantal delen rood plus het aantal delen blauw. Het aantal delen groen en rood mag maximaal 7 zijn, het aantal delen blauw is maximaal 3.

Sprite$-tabel:
Zie onder SCREEN 1.

Sprite-kleur-tabel:
Zie onder SCREEN 4.

Sprite-info-tabel:
Zie onder SCREEN 4.

Palette-tabel:
Zie onder SCREEN 0 (40-tekens).

Het besturen van de Video Display Processor gaat door middel van het laden cq uitlezen van de in de chip aanwezige registers. In de hiernavolgende beschrijving worden alle registers opgesomd. Daartoe zijn de registers gegroepeerd in functioneel bij elkaar behorende registers. Om de registers vanuit BASIC te kunnen beinvloeden, dient gebruik te worden gemaakt van de functie VDP(x).
De werkelijke registernummers en de nummers, die in de functie VDP moeten worden gebruikt, zijn niet altijd gelijk. Daarom zijn beide nummers in de tabel opgenomen. Zou u de registers vanuit een machinetaalroutine willen laden, dan dient u de registernummers te gebruiken (reg), wilt u de registers met VDP benaderen dan gebruikt u de VDP-nummers (VDP()).

MSX1 kent alleen de registers 0 tot en met 8.

VDP mode-registers:

VDP()	reg	7	6	5	4	3	2	1	0
0	0	0	DG	IE0	IE1	M5	M4	M3	D
1	1	-	BLK	IE2	M1	M2	0	SZ	MAG
9	8	MSE	LCS	TP	CBD	VRS1	VRS0	SPD	B/W
10	9	LN	0	SYM1	SYM0	IL	E/O	NTSC	DCD

```
DG          - Digitize
IE0         - Vertical Retrace Interrupt Enable
IE1         - Horizontal Retrace Interrupt Enable
IE2         - Light pen/mouse Interrupt Enable

M5/1        - M5 M4 M3 M2 M1
              --------------
              0  0  0  0  0  - SCREEN 1
              0  0  0  0  1  - SCREEN 0 (40-tekens)
              0  0  0  1  0  - SCREEN 3
              0  0  1  0  0  - SCREEN 2
              0  1  0  0  0  - SCREEN 4
              0  1  0  0  1  - SCREEN 0 (80-tekens)
              0  1  1  0  0  - SCREEN 5
              1  0  0  0  0  - SCREEN 6
              1  0  1  0  0  - SCREEN 7
              1  1  1  0  0  - SCREEN 8

D           - External VDP-input (0=ON) (Alleen MSX1)
BLK         - Enable/disable Display (1=Enable)
SZ          - Sprite size (0=8*8)
MAG         - Magnify sprites (1=vergroot)
MSE         - Light pen/mouse (1=muis)
LCS         - Light pen/coincidence select (1=light pen)
TP          - Transparant mode (0= code 0 is transparant)
                                1= code 0 niet transp.)
CBD         - Color Bus Direction (0=output)

VRS1/0      - Video Ram Select
              VRS1 VRS0
              ---------
                0    0     - 1 * 16 kByte
                0    1     - 4 * 16 kByte
                1    0     - 1 * 64 kByte
                1    1     - 64 kByte High Speed

SPD         - Sprite Disable
B/W         - Black and White mode (32 levels  Composite
              Video output)
```

SYM1/0 - Synchronisation Mode:

SYM1	SYM0	
0	0	- Intern
0	1	- Mix
1	0	- Extern (digitize)
1	1	- none

IL - Interlace mode (1=interlace)
E/O - Even/Odd
(0=normal display,
1=even/odd alternative screen display)
NTSC - TV-mode select (0=NTSC, 1=PAL)
DCD - Dot Clock Direction (0=dot clock input)

VDP adres-registers:

VDP()	reg	7	6	5	4	3	2	1	0
2	2	0	A16	A15	A14	A13	A12	A11	A10
3	3	B13	B12	B11	B10	B9	B8	B7	B6
11	10	0	0	0	0	0	B16	B15	B14
4	4	0	0	C16	C15	C14	C13	C12	C11
5	5	D14	D13	D12	D11	D10	D9	D8	D7
12	11	0	0	0	0	0	0	D16	D15
6	6	0	0	E16	E15	E14	E13	E12	E11
15	14	0	0	0	0	0	F16	F15	F14

A16/A10 - Scherm-info-tabel
B16/B6 - Kleur-tabel
C16/C11 - Matrix-tabel
D16/D7 - Sprite-info-tabel
E16/E11 - Sprite$-tabel
F16/F14 - Video Ram Acces

VDP tekstcontrole-registers:

VDP()	reg	7	6	5	4	3	2	1	0
7	7	TC3	TC2	TC1	TC0	BDC3	BDC2	BDC1	BDC0
13	12	C3	C2	C1	C0	BC3	BC2	BC1	BC0
14	13	ON3	ON2	ON1	ON0	OF3	OF2	OF1	OF0

TC3/0 - Text Color
BCD3/0 - Back Drop Color
C3/0 - Color
BC3/0 - Back Color
ON3/0 - Blink On (aantal perioden van 1/5 seconde)
OF3/0 - Blink Off (aantal perioden van 1/5 seconde)

VDP controle-registers:

VDP()	reg	7	6	5	4	3	2	1	0
16	15	0	0	0	0	RS3	RS2	RS1	RS0
17	16	0	0	0	0	C3	C2	C1	C0
18	17	0	0	RC5	RC4	RC3	RC2	RC1	RC0
19	18	dV3	dV2	dV1	dV0	dH3	dH2	dH1	dH0
20	19	IL7	IL6	IL5	IL4	IL3	IL2	IL1	IL0
21	20	0	0	CBX5	CBX4	CBX3	CBX2	CBX1	CBX0
22	21	0	0	CBY5	CBY4	CBY3	CBY2	CBY1	CBY0
23	22	0	0	CBZ5	CBZ4	CBZ3	CBZ2	CBZ1	CBZ0

RS3/0 - Registernummer van Status-register
C3/0 - Color code (relatief palette-adres)
RC5/0 - Registernummer van Control-register
dV3/0 - delta Vertical adjust.
dH3/0 - delta Horizontal adjust.
IL7/0 - verticale line number of Line Interrupt.
CBX5/0 - color burst value of phase 0
CBY5/0 - color burst value of phase 1/3
CBZ5/0 - color burst value of phase 2/3

out 09Ah,||rood|blauw||
out 09Ah,|| - |groen||

VDP status-registers:

VDP()	reg	7	6	5	4	3	2	1	0
8	0	F	SD	C	S4	S3	S2	S1	S0
-1	1	FL	LPS	I4	I3	I2	I1	I0	FH
-2	2	TR	VR	HR	BD	0	0	E/O	CE
-3	3	X7	X6	X5	X4	X3	X2	X1	X0
-4	4	0	0	0	0	0	0	X9	X8
-5	5	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
-6	6	0	0	0	0	0	0	Y9	Y8
-7	7	C7	C6	C5	C4	C3	C2	C1	C0
-8	8	BX7	BX6	BX5	BX4	BX3	BX2	BX1	BX0
-9	9	0	0	0	0	0	0	BX9	BX8

F - Vertical retrace interrupt. Wordt gereset door het uitlezen van dit register (reg. 0).
SD - Vijfde cq. negende sprite gedetecteerd. Wordt gereset door het uitlezen van dit register.
C - Botsing tussen sprites geconstateerd.
S4/0 - Nummer van de vijfde cq. negende sprite.
FL - Muis of lichtpen schakelaar status.
LPS - Tweede muis of lichtpen schakelaar status.
I4/0 - LSI-versie nummer.
FH - Horizontal retrace interrupt. Wordt gereset door het uitlezen van dit register (reg. 1).
TR - Transfer met CPU data Ready.
VR - Vertical Retrace timing.
HR - Horizontal Retrace timing.
BD - Border Detected.
E/O - Even or Odd field status.
CE - Command Executing status (0=ready)
X9/0 - X-coordinaat van sprite-coincidentie, muis of lichtpen.
Y9/0 - Y-coordinaat van sprite-coincidentie, muis of lichtpen.
C7/0 - Kleurencode van gelezen kleurregister.
BX9/0 - Border X-coordinaat van sprite-coinciden-tie, muis of lichtpen.

VDP programma-registers:

VDP()	reg	7	6	5	4	3	2	1	0
33	32	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0
34	33	0	0	0	0	0	0	0	SX8
35	34	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0
36	35	0	0	0	0	0	0	SY9	SY8
37	36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0
38	37	0	0	0	0	0	0	0	DX8
39	38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0
40	39	0	0	0	0	0	0	DY9	DY8
41	40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0
42	41	0	0	0	0	0	0	NX9	NX8
43	42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0
44	43	0	0	0	0	0	0	NY9	NY8
45	44	CH3	CH2	CH1	CH0	CL3	CL2	CL1	CL0
46	45	0	MXC	MXD	MXS	Y	X	E/N	M/M
47	46	CM3	CM2	CM1	CM0	LO3	LO2	LO1	LO0

SX8/0 - Source X-coordinaat
SY9/0 - Source Y-coordinaat
DX8/0 - Destination X-coordinaat
DY9/0 - Destination Y-coordinaat
NX9/0 - Aantal horizontale pixels
NY9/0 - Aantal verticale pixels
CH3/0 - Kleurcode
CL3/0 - Kleurcode
MXC - CPU-acces external memory
MXD - Destination external memory
MXS - Source external memory
Y - Richting Y-coordinaat
X - Richting X-coordinaat
E/N - Equal/Not equal detected (0=equal)
M/M - Major/Minor (0=X-as is major richting
1=Y-as is major richting)

CM3/0 - Commando:

CM3	CM2	CM1	CM0	Bewerking:	Richting:
1	1	1	1	Move byte	CPU > VRAM
1	1	1	0	Move byte	VRAM > CPU
1	1	0	1	Move byte	VRAM > VRAM
1	1	0	0	Move byte	VDP > VRAM
1	0	1	1	Move dot	CPU > VRAM
1	0	1	0	Move dot	VRAM > CPU
1	0	0	1	Move dot	VRAM > VRAM
1	0	0	0	Move dot	VDP > VRAM
0	1	1	1	Line dot	VDP > VRAM
0	1	1	0	Search dot	VRAM > VDP
0	1	0	1	PSET dot	VDP > VRAM
0	1	0	0	POINT dot	VRAM > VDP

LO3/0 - Logische Operatie:

LO3	LO2	LO1	LO0	Bewerking:
0	0	0	0	SC -> DC
0	0	0	1	SC AND DC -> DC
0	0	1	0	SC OR DC -> DC
0	0	1	1	SC XOR DC -> DC
0	1	0	0	NOT SC -> DC
1	0	0	0	IF SC=0 AND TP=0 THEN DC -> DC ELSE SC -> DC
1	0	0	1	IF SC=0 AND TP=0 THEN DC -> DC ELSE SC AND DC -> DC
1	0	1	0	IF SC=0 AND TP=0 THEN DC -> DC ELSE SC OR DC -> DC
1	0	1	1	IF SC=0 AND TP=0 THEN DC -> DC ELSE SC XOR DC -> DC
1	1	0	0	IF SC=0 AND TP=0 THEN DC -> DC ELSE NOT SC -> DC

35 Programma's

Kleurcode in scherm-info-tabel onder screen8

In SCREEN 8 bevat de scherm-info-tabel voor ieder pixel een byte, waarin de kleur van dat byte is aangegeven. Er wordt echter geen gebruik gemaakt van een kleurnummer, via hetwelk de intensiteit van de kleuren rood, groen en blauw uit de palette-tabel wordt gelezen. In plaats daarvan is ieder byte samengesteld uit de intensiteitswaarden van ieder van de kleuren rood, groen en blauw.

Het volgende programma vraagt u de intensiteit voor iedere basiskleur op te geven, waarna de waarde van het byte in de scherm-info-tabel wordt berekend (zie regel 130). Met de geneste FOR/NEXT-lus wordt vervolgens een blokje in de door u opgegeven kleur in het Video-RAM ge-POKEd. U ziet het resultaat op het scherm verschijnen.

Regel 170 laat meteen zien volgens welke formule het adres in de scherm-info-tabel voor een bepaald pixel kan worden gevonden (de horizontale positie plus 256 keer de verticale positie).

```
100 INPUT "Intensiteit van ROOD  (0-7)";R
110 INPUT "Intensiteit van GROEN (0-7)";G
120 INPUT "Intensiteit van BLAUW (0-3)";B
130 KL=32*G+4*R+B
140 SCREEN 8
150 FOR X=100 TO 110
160 FOR Y=100 TO 110
170 VPOKE BASE(40)+X+256*Y,KL
180 NEXT Y
190 NEXT X
200 IF INKEY$="" THEN GOTO 200
210 GOTO 100
```

Beeldschermpagina's in video-RAM

Het volgende programma laat zien hoe een andere pagina, dan de pagina waarop wordt getekend, kan worden weergegeven. Terwijl pagina 0 op het beeldscherm staat, worden pagina's 1 en 2 met tekeningen gevuld. Daarna kunt u door middel van het intoetsen van het gewenste paginanummer een van de pagina's 0, 1 of 2 op het scherm afdrukken. Het programma is erg eenvoudig, doch het laat de verbazingwekkende snelheid zien, waarmee de schermen achtereenvolgens zichtbaar kunnen worden gemaakt. Heeft uw computer slechts 64 kbytes videogeheugen, dan kunt u maximaal 2 pagina's (0 en 1) in het geheugen kwijt.

```
10 INPUT "voorgrondkleur  ";VK
20 COLOR VK
30 SCREEN 5
40 SET PAGE 0,0
50 OPEN "grp:" AS #1
60 PSET(0,0):PRESET(0,0)
70 PRINT #1,"DIT IS SCHERM PAGINA 0"
80 PSET(0,20):PRESET(0,20)
90 PRINT #1,"TEKEN CIRKELS IN PAGINA 1"
100 SET PAGE 0,1
110 FOR I=1 TO 10
120 CIRCLE (127,100),RND(1)*100,VK
130 NEXT I
140 SET PAGE 0,0
150 PSET(0,40):PRESET(0,40)
160 PRINT #1,"TEKEN BLOKKEN IN PAGINA 2"
170 SET PAGE 0,2
180 FOR I=1 TO 10
190 X=RND(1)*200:Y=RND(1)*150
200 LINE (X,Y)-(X+50,Y+50),VK,B
210 NEXT I
220 SET PAGE 0,0
230 PSET(0,60):PRESET(0,60)
240 PRINT #1,"WELKE PAGINA WILT U ZIEN?"
250 N$=INKEY$: IF N$="" THEN GOTO 250
260 IF N$<"0" OR N$>"2" THEN GOTO 240
270 N=VAL(N$)
280 SET PAGE N
290 GOTO 250
```

Geheugen-dump

Wat staat er nou eigenlijk in het geheugen? Op deze vraag krijgt u gauw antwoord, wanneer u het volgende programma intikt en uitvoert. Het laat u precies zien wat er op een door u opgegeven plaats in het geheugen staat, zowel in hexadecimale- als in ASCII-vorm.

Rekening houdend met de maximale regellengte op het beeldscherm, van 40 karakters, kwam ik tot de volgende verdeling:

4 posities voor het geheugenadres (hexadecimaal).
2 spaties.
16 posities voor de hexadecimale weergave van de inhoud van 8 opeenvolgende geheugenadressen.
2 spaties.
8 posities voor de ASCII-weergave van dezelfde 8 opeenvolgende geheugenadressen.

In totaal zijn dit maar 32 karakters per regel. Het is echter, in verband met het tellen en het relateren aan geheugenadressen, handig om in groepjes van 8 te werken. Nog handiger zouden groepjes van 16 zijn, echter, dat past niet op een regel.

Met regel 80 tot en met 110 worden het begin en eind-adres van het te dumpen geheugengebied opgevraagd. Op regel 120 start een FOR-NEXT lus die voor iedere regel op het scherm eenmaal wordt doorlopen. Binnen die lus worden de volgende acties ondernomen.

Converteer het decimale adres van de eerste, op de regel af te drukken, geheugenlocatie naar hexadecimaal en zet dit in P$ (regels 130 tot en met 190, met gebruikmaking van regel 70). Voer hierna de FOR-NEXT lus van regel 200 acht maal uit. In deze lus worden 8 geheugenlocaties uitgelezen en in hexadecimaal formaat in P$ gezet. Voer daarna de FOR-NEXT lus van regel 260 acht maal uit. In deze lus worden dezelfde geheugenlocaties nogmaals uitgelezen, doch nu worden ze als ASCII codes in P$ gezet. Hierbij dient nog te worden opgemerkt dat "unprintable" karakters worden vervangen door een "." (regel 280). Print nu de variabele P$ uit. Hiermee is een regel op het beeldscherm gezet.

```
10 '****************************
20 '*  Afdrukken van de inhoud  *
30 '*  van het geheugen. Start  *
40 '*  en eindadres op te geven *
50 '****************************
60 '
70 H$="0123456789ABCDEF"
80 INPUT "Startadres";S
90 IF S<0 THEN S=S+2^16
100 INPUT "Eindadres";E
110 IF E<0 THEN E=E+2^16
120 FOR I=S TO E STEP 8
130 H1=INT(I/4096)
140 R=I-H1*4096
150 H2=INT(R/256)
160 R=R-H2*256
170 H3=INT(R/16)
180 H4=R-H3*16
190 P$=MID$(H$,H1+1,1)+MID$(H$,H2+1,1
)+MID$(H$,H3+1,1)+MID$(H$,H4+1,1)+"
"
200 FOR J=1 TO 8
210 X$=HEX$(PEEK(I+J))
220 IF LEN(X$)<2 THEN X$="0"+X$
230 P$=P$+X$
240 NEXT J
250 P$=P$+"  "
260 FOR J=1 TO 8
270 X$=CHR$(PEEK(I+J))
280 IF ASC(X$)<32 THEN X$="."
290 P$=P$+X$
300 NEXT J
310 PRINT P$
320 NEXT I
330 PRINT "NOG MEER TE DUMPEN? (J/N)"
340 I$=INKEY$
350 IF I$<>"J" AND I$<>"N" AND I$<>"j
" AND I$<>"n" GOTO 340
360 IF I$="J" OR I$="j" GOTO 80
370 END
```

Met het vorige programma is al veel mogelijk. Het leek mij echter interessant om een wat georganiseerde geheugen-dump van het BASIC-geheugen te maken. Het volgende programma doet dat.

De regelnummers zijn nogal hoog gekozen. De reden hiervan is, dat je dit programma achter het te onderzoeken programma moet kunnen plaatsen. Door middel van een GOTO 10000 commando wordt het dump-programma dan gestart, waarbij het een dump zal maken van alle programmaregels die een lager regelnummer hebben dan 10000.

Op regel 10040 wordt het beginadres van het BASIC-geheugen bepaald. Dat adres wordt met regel 10050 op het beeldscherm afgedrukt. De eerste BASIC-regel start op dat adres. Iedere BASIC-regel heeft een vast formaat dat er als volgt uitziet:

```
    adres van
 volgende regel  regelnummer      regelinhoud

    lsb    msb    lsb    msb
|------|------|------|------|---------//---------|
```

De eerste positie is het minst significante byte van het adres waar de volgende BASIC-regel begint. De tweede positie is het meest significante byte van dat adres. Met regels 10060 tot en met 10080 wordt het adres van de volgende regel bepaald.

De derde en vierde positie bevatten respectievelijk het minst en meest significante byte van het regelnummer van de huidige regel. Dit regelnummer wordt op de regels 10100 tot en met 10120 bepaald. Vervolgens wordt met regel 10130 onmiddellijk gecontroleerd of de huidige regel soms regel 10000 is. In dat geval wordt het programma beeindigd.

Is de laatste regel nog niet bereikt, dan wordt het regel nummer afgedrukt, waarna met regels 10150 tot en met 10170 de inhoud van de BASIC-regel hexadecimaal wordt afgedrukt. Voordat de volgende regel wordt behandeld, zorgt regel 10190 er voor dat het adres van die volgende regel in de variabele P komt (P=pointer).

```
10000 '*************************
10010 '* BASIC GEHEUGEN DUMPEN *
10020 '*************************
10030 '
10040 P=2^15+1
10050 PRINT "ADRES=";P
10060 VR=PEEK(P)
10070 P=P+1
10080 VR=VR+256*PEEK(P)
10090 P=P+1
10100 RN=PEEK(P)
10110 P=P+1
10120 RN=RN+256*PEEK(P)
10130 IF RN=>10000 GOTO 10210
10140 PRINT "REGEL=";RN
10150 FOR A=P+1 TO VR-1
10160 PRINT USING "\ \";HEX$(PEEK(A));
10170 NEXT A
10180 PRINT: PRINT
10190 P=VR
10200 GOTO 10050
10210 PRINT "EINDE BASIC LISTING"
10220 END
```

Naar aanleiding van het voorgaande programma kwam het idee bij me op om, nu het formaat van een BASIC-programma bekend is, daar nog iets meer mee te doen. Door binnen de inhoud van een BASIC-regel naar het al of niet voorkomen van een bepaalde string te zoeken, kun je bijvoorbeeld eenvoudig bepalen of een bepaalde variabele of een bepaalde tekst in het programma voorkomt, en zo ja, in welke regel(s).

Zoals u in de listing ziet, zit een groot deel van het vorige programma in dit programma verwerkt. Met regel 10050 wordt de te zoeken string gevraagd en in variabele T$ gezet. Regel 10060 bepaalt de lengte van de ingegeven string.

Nu wordt het begin van de eerste BASIC-regel opgezocht. Eenmaal gevonden, wordt er met regel 10200 gekeken of het eerste karakter in de BASIC-regel gelijk is aan het eerste karakter in de opgegeven string. Is dat inderdaad het geval, dan wordt gekeken of het tweede karakter ook gelijk is aan het tweede karakter uit de opgegeven string. Dit gaat zo door tot er een ongelijkheid optreedt of totdat alle karakters van de string in de regel blijken voor te komen. In het eerste geval zal de variabele N gelijk aan 1 worden gemaakt. In het laatste geval zal de variabele N zijn in regel 10200 verkregen waarde houden.

In regel 10210 wordt vervolgens gekeken of N groter is dan het aantal karakters in de opgegeven string. Is dat zo, dan is de string dus gevonden, en wordt het regelnummer van de BASIC-regel waarin die string was gevonden afgedrukt. Hierna wordt de volgende BASIC-regel bekeken en herhaalt zich het hele proces.

Binnen dat proces zit nog regel 10160. Hiermee wordt gecontroleerd of het regelnummer van de te onderzoeken BASIC-regel wel lager is dan 10000. Net als bij het vorige programma is het ook hier weer de bedoeling om het zoekprogramma samen met het te onderzoeken programma in het geheugen te zetten. Het zoekprogramma wordt dan gestart door een GOTO-commando te geven.

Nadat de laatste regel van het te onderzoeken programma is verwerkt, geven de regels 10270 tot en met 10300 de gelegenheid om opnieuw te beginnen, eventueel met een andere string.

```
10000 '*************************
10010 '* TEKST ZOEKEN IN BASIC *
10020 '*************************
10030 '
10040 SCREEN 0: WIDTH 36
10050 INPUT "VAN WELKE TEKST WILT U
WETEN OF DIE IN HET PROGRAMMA VOORKO
MT";T$
10060 L=LEN(T$)
10070 P=2^15+1
10080 VR=PEEK(P)
10090 P=P+1
10100 VR=VR+256*PEEK(P)
10110 IF VR<=0 GOTO 10270
10120 P=P+1
10130 RN=PEEK(P)
10140 P=P+1
10150 RN=RN+256*PEEK(P)
10160 IF RN=10000 GOTO 10270
10170 P=P+1
10180 N=1
10190 FOR A=P TO VR-1
10200 IF MID$(T$,N,1)=CHR$(PEEK(P))
THEN N=N+1 ELSE N=1
10210 IF N>L THEN PRINT USING "#####
#";RN;
10250 P=VR
10260 GOTO 10080
10270 PRINT: PRINT "ANDERE TEKSTEN Z
OEKEN? (J/N)"
10280 I$=INKEY$
10290 IF I$<>"J" AND I$<>"j" AND I$<
>"N" AND I$<>"n" GOTO 10280
10300 IF I$="J" OR I$="j" GOTO 10050
10310 END
```

Inhoudsopgave van de schijf

Het volgende programma is weliswaar erg eenvoudig, maar laat toch een aantal leuke mogelijkheden van het MSX-BASIC zien. Bovendien is het mijn ervaring dat het nog een erg handig programmaatje is ook. Wat het namelijk doet is, dat het alle op de schijf staande bestanden en programma's op het beeldscherm afdrukt. Vervolgens geeft het aan hoeveel kilobytes vrije ruimte er nog op de schijf is. Daarna vraagt het u welk programma u wilt draaien.

Het is een goede gewoonte om BASIC programma's op schijf op te slaan onder een vrij te kiezen naam, met de toevoeging ".BAS". Hierdoor herkent u de BASIC programma's onmiddellijk, en zult u niet gauw per ongeluk een bestand proberen te starten. Aan bestandsnamen kunt u bijvoorbeeld ".DAT" toevoegen.

Op de vraag, welk programma u wilt draaien, hoeft u echter de toevoeging ".BAS" niet in te tikken. Deze toevoeging is al in het programma opgenomen (zie regel 1140. Bovendien zorgt deze geprogrammeerde toevoeging er voor dat u niets anders dan programma's met de toevoeging ".BAS" kunt starten. Hiermee worden vergissingen dus tegengegaan.

De eigenlijke inhoudsopgave van de schijf wordt met regel 1080 gemaakt. Met de functie DSKF(n) wordt de nog vrije ruimte op de schijf bepaald. De reden om een beeldschermbreedte van 38 karakters te kiezen (regel 1040), is dat er dan precies drie file-namen op een regel gaan.

Door dit programma onder de naam "AUTOEXEC.BAS" op de schijf te zetten, zal dit programma na het aanschakelen van de computer (en de schijf) automatisch worden gestart. Dit heeft dan tot gevolg dat u direct na het aanschakelen, zonder enige verdere handelingen, kunt zien welke programma's er op de schijf staan. Vervolgens kunt u dan door eenvoudigweg de programmanaam in te tikken, het gewenste programma laden.

```
1000 '**********************
1010 '*    AUTOEXEC.BAS    *
1020 '**********************
1030 '
1040 WIDTH(38)
1050 CLS
1060 PRINT "OVERZICHT VAN BESTANDEN EN
 PROGRAMMA'S"
1070 PRINT "**************************
************"
1080 FILES
1090 PRINT
1100 PRINT "**************************
************"
1110 PRINT "VRIJE RUIMTE =";DSKF(0);"K
."
1120 PRINT
1130 INPUT "WELK PROGRAMMA WILT U DRAA
IEN";P$
1140 LOAD P$+".BAS"
```

System-reset via toetsencombinatie

Sommige MSX-computers hebben geen reset-toets. Om die computers te kunnen resetten moet de computer uit- en weer aangeschakeld worden. Er is echter een andere mogelijkheid en die wordt in het volgende programma gegeven.

Het programma laadt een klein machinetaalroutinetje in het geheugen en wijzigt vervolgens het hook-adres dat 50 keer per seconde wordt aangeroepen (timer-interrupt), zodanig, dat het zojuist geladen machinetaalroutinetje wordt uitgevoerd, elke keer dat het hook-adres wordt aangeroepen.

De machinetaalroutine leest de keyboard-matrix en controleert of de toetsen CAPS, GRAPH, CONTROL en SHIFT alle vier tegelijkertijd zijn ingedrukt. Is dat het geval, dan wordt er naar adres 0 uit de ROM gesprongen. Dit is hetzelfde adres dat ook wordt aangeroepen na het aanschakelen van de computer. Het gevolg is dan ook, dat het gehele geheugen wordt ge-initialiseerd. U hebt de computer dus gereset, zonder hem aan of uit te schakelen.

```
1000 '*********************************
1010 '* DIT PROGRAMMA ZET OP HOOKADRES*
1020 '* "TIMI" EEN SPRONG-INSTRUCTIE  *
1030 '* NAAR MC-ROUTINE, WAARMEE DE   *
1040 '* TOETSENCOMBINATIE:            *
1050 '*      CAP/GRAPH/CONTROL/SHIFT  *
1060 '* WORDT GECONTROLEERD.          *
1070 '* IS DEZE COMBINATIE INGEDRUKT, *
1080 '* DAN WORDT COMPUTER GE-RESET.  *
1090 '*********************************
1100 '
1110 CLEAR 200,&HCFFF
1120 RESTORE
1130 FOR I=0 TO 10
1140 READ MC$: POKE &HD000+I,VAL("&H"+MC$)
1150 NEXT I
1160 'SET HOOK-ADRES "TIMI"
1170 POKE &HFD9F+2,&HD0
```

```
1180 POKE &HFD9F+1,&H0
1190 POKE &HFD9F,&HCD
1200 END
1210 'CHECK TOETSEN-COMBINATIE
1220 DATA 3E,06:'     LD   A,<rij-nr.6>
1230 DATA CD,41,01:'  CALL SNSMAT
1240 DATA FE,F0:'     CP   '11110000'
1250 DATA CA,00,00:   JP   Z,CHKRAM
1260 DATA C9:'        RET
```

Sectoren van schijf inlezen

Het volgende programma geeft een voorbeeld van het aanroepen van BDOS-functies. We gebruiken daartoe een klein machinetaalprogramma, waarmee we Z80-registers laden met de door ons gewenste inhoud (functie-nummer etc.), waarna we het adres &HF37D aanroepen.

Met de subroutine van regel 270 wordt de machinetaal routine in het vrijgemaakte geheugendeel geladen. Op regel 170 wordt de machinetaalroutine aangeroepen. De machinetaalroutine zet de gelezen sector in het RAM-geheugen, vanaf adres &HC000. Met de regels 180 tot en met 200 wordt de inhoud van de gelezen sector in rijtjes van 16 bytes afgedrukt naar een printer.

In de machinetaalroutine (zie DATA-regels 400 en 410) zijn vaste waarden opgenomen voor de te lezen drive en sector. U zou deze waarden aan de operator kunnen vragen en invullen in deze instructies, voordat u werkelijk overgaat tot lezen. De eenvoudigste manier is echter het gewenste nummer in deze routine in te vullen voordat u hem uitvoert.

In plaats van lezen kunt u ook een sector naar de schijf schrijven. Om dat te doen dient u de waarde &H2F uit DATA-regel 420 te vervangen door de waarde &H30.

```
100 '*********************************
110 '*      LEES SECTOREN VAN DISK   *
120 '*********************************
130 '
140 WIDTH 39:CLEAR 1000,&HB7FF
150 DEFUSR0=&HB800
160 GOSUB 270:'LADEN MACHINECODE
170 DUMMY=USR0(I)
180 FOR I=0 TO 15
185 FOR J=0 TO 15
190 LPRINT RIGHT$("00"+HEX$(PEEK(&HC000+I*16+J)),2);" ";
194 NEXT J
196 LPRINT
200 NEXT I
210 END
```

```
220 '
230 '********************************
240 '* SUBROUTINE: LADEN MACHINECODE *
250 '********************************
260 '
270 FOR I=0 TO 20
280 READ H$:POKE &HB800+I,VAL("&H"+H$)
290 NEXT I
300 RETURN
310 '
320 '********************************
330 '*     MACHINETAAL PROGRAMMA     *
340 '********************************
350 '
360 DATA 11,00,C0:' TRANSFERADDR.-> DE
370 DATA 0E,1A:'    SET DMA MODE
380 DATA CD,7D,F3:' CALL SYSTEM
390 DATA 26,01:'    AANTAL SECTOREN -> H
400 DATA 2E,00:'    DRIVE NUMMER -> L
410 DATA 11,00,00:' SECTOR NUMMER -> DE
420 DATA 0E,2F:'    LEZEN (30=SCHRIJVEN)
430 DATA CD,7D,F3:' CALL SYSTEM
440 DATA C9:'       RETURN
```

Controleren van de printerstatus

De meest eenvoudige manier om de status van de printer op te vragen is als volgt:

```
10000 IF INP(&H90)=122 THEN PRINT "Maak printer gereed"
10001 IF INKEY$<>CHR$(13) THEN GOTO 10000
```

Hierbij wordt echter direct een I/O-poort geadresseerd. Niet alle fabrikanten gebruiken echter dezelfde poortnummers voor dezelfde poorten. Het is daarom beter om gebruik te maken van een BIOS-entry point, waarmee de routine die de status van de printer opvraagt wordt aangeroepen.

Het hiernavolgende voorbeeld laat zien, hoe het eigenlijk hoort. Er wordt gebruik gemaakt van een BIOS-entry point om de printer-poort te benaderen.

Het programma bestaat uit twee delen, een deel waarmee een machinetaalroutinetje in het vrijgemaakte deel van het geheugen wordt gezet (regels 32000 tot en met 32110) en een deel, waarmee het machinetaalroutinetje wordt aangeroepen (vanaf regel 32190 tot en met 32240).

Het eerste deel dient eenmaal, na aanschakelen van de computer te worden geladen. Het tweede deel geeft een manier aan, waarop de machinetaalroutine kan worden aangeroepen.

Indien de printer niet is aangesloten, zult u worden gevraagd de printer alsnog aan te zetten en aan te sluiten, alvorens het programma verder kan gaan. Deze procedure dient steeds, voordat er moet worden geprint, worden uitgevoerd. U zult dan nooit het idee krijgen dat uw systeem hangt, terwijl het systeem alleen maar wacht tot het zijn data kwijt kan aan de printer.

```
1 CLEAR 200,&HCFFF
2 GOSUB 32000
3 LPRINT "programma werkt"
4 END
```

De subroutine staat op de volgende pagina.

```
31900 '******************************
31910 '* MACHINETAALROUTINE:         *
31920 '* USR0 = CHECK PRINTER STATUS *
31930 '*        (KABEL/READY/PAPIER) *
31940 '******************************
31950 '
32000 RESTORE 32060
32010 FOR I=1 TO 12
32020 READ MC$
32030 POKE &HCFFF+I,VAL("&H"+MC$)
32040 NEXT I
32050 '*** LINE PRINTER STATUS ***
32060 DATA CD,A8,00:'CALL LPTSTT
32070 DATA 32,F8,F7:'LD   (nn),A
32080 DATA 3E,02:   'LD   A,2
32090 DATA 32,63,F6:'LD   (nn),A
32100 DATA C9:      'RET
32110 DEFUSR0=&HD000
32120 '
32130 '*****************************
32140 '* BASIC-PROGRAMMA, DAT HET   *
32150 '* AFDRUKKEN VOORKOMT, ZOLANG *
32160 '* DE PRINTER NIET READY IS   *
32170 '*****************************
32180 '
32190 STATUS%=USR0(0)
32200 IF STATUS%=255 THEN RETURN
32210 PRINT "Maak de printer READY."
32220 PRINT "Druk daarna op RETURN."
32230 IF INKEY$<>CHR$(13) THEN GOTO 32230
32240 GOTO 32190
```

Gemakkelijk starten van de programma's

Met het volgende programma kunnen programma's, door ze eenvoudigweg aan te wijzen, worden gestart. Daar het programma geheel in BASIC is geschreven, is verdere uitleg niet nodig.

```
10 'Met dit programma wordt een over-
20 'zicht van alle op schijf staande
30 'files gegeven. Vervolgens kan een
40 'file met de cursor worden aange-
50 'wezen.
60 'Door op RETURN te drukken, wordt
70 'de naam van de aangewezen file in
80 'F$ gezet, en indien het een BASIC
90 'programma is, uitgevoerd.
100 '
110 SCREEN 0:WIDTH 40:KEY OFF:CLS:X=0:Y=0:LOCATE X,Y:FI
LES
120 PRINT:PRINT:PRINT"Wijs de gewenste file aan met de
cursor.Druk vervolgens op RETURN."
130 LOCATE X,Y,1:I$=INKEY$
140 IF I$=CHR$(30) THEN Y=Y-1:IF Y<0 THEN Y=0
150 IF I$=CHR$(28) THEN X=X+13:IF X>26 THEN X=0:Y=Y+1:I
F Y>20 THEN Y=20
160 IF I$=CHR$(31) THEN Y=Y+1:IF Y>20 THEN Y=20
170 IF I$=CHR$(29) THEN X=X-13:IF X<0 THEN X=26:Y=Y-1:I
F Y<0 THEN Y=0
180 IF I$<>CHR$(13) THEN 130
190 LOCATE 0,23,0:F$=""
200 FOR I=0 TO 11
210 F$=F$+CHR$(VPEEK(BASE(0)+X+I+Y*40))
220 NEXT I
230 PT$=RIGHT$(F$,3)
240 IF PT$="BAS" OR PT$="bas" THEN RUN F$
250 PRINT "Dit is geen BASIC-programma";
260 FOR I=1 TO 500:NEXT I
270 GOTO 110
```

Het gebruik van systeemlocatie LINLEN

Het volgende korte programma laat zien wat er gebeurt wanneer de regellengte wordt gewijzigd door alleen de systeemlocatie LINLEN met een nieuwe regellengte te laden. Probeert u zelf maar eens een aantal verschillende waarden.

```
10 POKE 62384!,40
20 FOR I=1 TO 20
30 PRINT "........................................";
40 NEXT I
50 POKE 62384!,20
60 FOR I=1 TO 20
70 PRINT "01234567890123456789";
80 NEXT I
90 POKE 62384!,40
100 END
```

Hardcopy machinetaalroutines

Met de volgende twee programma's, die geheel in BASIC zijn geschreven, worden machinetaalroutines, waarmee een afdruk van het beeldscherm (opgebouwd onder SCREEN 2) naar een printer kan worden gemaakt.

Dat er twee routines zijn opgenomen, heeft te maken met kleine verschillen tussen printers onderling. Het eerste voorbeeld is bestemd voor een MSX-printer. Dit werkt voor alle matrix-printers, die aan de MSX-standaard voor printers voldoen. Het tweede programma is geschreven voor een Seikosha SP800 printer, maar werkt ook, zonder enige wijziging op Epson printers.

Met regel 1000 wordt een deel van het RAM-geheugen vrijgemaakt voor opslag van eigen machinetaalroutines. Met regels 1010 tot en met 1030 wordt vervolgens de machinetaalroutine, die in de DATA-regels vanaf regel 1070 is opgeslagen, in het vrijgemaakte geheugendeel gezet. Vervolgens wordt de machinetaalroutine een nummer gegeven met regel 1040. Vanaf dit moment kan de machinetaalroutine met de functie USR0(0) worden aangeroepen. Het BASIC-programma is nu ook niet meer nodig. U kunt het uit het geheugen verwijderen, of overschrijven met een ander programma.

Draait u nu een programma, waarmee in SCREEN 2 een tekening op het scherm wordt gemaakt, en u wilt die tekening op de printer afdrukken, dan kunt u in dat programma de functie USR0(0) opnemen. Op het moment dat de funtie wordt uitgevoerd, zal het beeldscherm op de printer worden afgedrukt, waarna het programma weer verder gaat.

Het is ook mogelijk het maken van een "hard-copy" te starten op het indrukken van een functietoets. Daartoe is in het hiernavolgende programma regel 1050 opgenomen. Daarmee wordt functietoets 5 de hardcopy-toets. Het programma waarbinnen u de hardcopy wilt gaan maken, moet het indrukken van de hardcopytoets wel toestaan.

Tenslotte dient nog te worden opgemerkt, dat in de DATA-statements niet alleen data zijn opgenomen, maar

ook commentaar. U mag dit commentaar (alles van de dubbele punt tot het einde van de regel) gewoon weglaten. Dat scheelt flink in tikwerk. Bovendien mag u meerdere data-items in een DATA-regel zetten. Op die manier kan het aantal DATA-regels sterk worden ingekrompen en bespaart u zich nog meer tikwerk.

```
1 '*********************************
2 '*  HARDCOPY ROUTINE VAN SCREEN 2  *
3 '*       NAAR EEN MSX-PRINTER.     *
4 '*********************************
5 '
1000 CLEAR 200,&HBFFF
1010 FOR I=0 TO 215
1020 READ MC$:POKE &HC000+I,VAL("&H"+MC$)
1030 NEXT I
1040 DEFUSR0=&HC000
1050 KEY5,"GRPRI=USR0(0)"+CHR$(13)
1060 END
1070 DATA 3E,1B:'                LD   A,1B
1080 DATA CD,A5,00:'             CALL LPTOUT
1090 DATA 3E,42:'                LD   A,42
1100 DATA CD,A5,00:'             CALL LPTOUT
1110 DATA 00,00:'                NOP
1120 DATA 00,00,00:'             NOP
1130 DATA 21,00,00:'             LD   HL,0
1140 DATA 06,18:'                LD   B,24
1150 DATA C5:'          REGEL:   PUSH BC
1160 DATA CD,AA,C0:'             CALL SETGR:
1170 DATA 11,D8,C0:'             LD   DE,BUFFER:
1180 DATA 01,00,01:'             LD   BC,#0100
1190 DATA CD,59,00:'             CALL LDIRMV
1200 DATA 11,00,01:'             LD   DE,#0100
1210 DATA 19:'                   ADD  HL,DE
1220 DATA E5:'                   PUSH HL
1230 DATA 21,D8,C0:'             LD   HL,BUFFER:
1240 DATA 06,20:'                LD   B,32
1250 DATA C5:'          TEKEN:   PUSH BC
1260 DATA 0E,08:'                LD   C,8
1270 DATA 11,07,00:' KOLOM:      LD   DE,7
1280 DATA 06,00:'                LD   B,0
1290 DATA 7E:'                   LD   A,(HL)
1300 DATA CB,07:'                RLC  A
1310 DATA 30,02:'                JR   NC,L6:
1320 DATA CB,C0:'                SET  0,B
```

```
1330 DATA 77:'          L6:     LD   (HL),A
1340 DATA 23:'                  INC  HL
1350 DATA 7E:'                  LD   A,(HL)
1360 DATA CB,07:'               RLC  A
1370 DATA 30,02:'               JR   NC,L5:
1380 DATA CB,C8:'               SET  6,B
1390 DATA 77:'          L6:     LD   (HL),A
1400 DATA 23:'                  INC  HL
1410 DATA 7E:'                  LD   A,(HL)
1420 DATA CB,07:'               RLC  A
1430 DATA 30,02:'               JR   NC,L4:
1440 DATA CB,D0:'               SET  5,B
1450 DATA 77:'          L4:     LD   (HL),A
1460 DATA 23:'                  INC  HL
1470 DATA 7E:'                  LD   A,(HL)
1480 DATA CB,07:'               RLC  A
1490 DATA 30,02:'               JR   NC,L3:
1500 DATA CB,D8:'               SET  4,B
1510 DATA 77:'          L3:     LD   (HL),A
1520 DATA 23:'                  INC  HL
1530 DATA 7E:'                  LD   A,(HL)
1540 DATA CB,07:'               RLC  A
1550 DATA 30,02:'               JR   NC,L2:
1560 DATA CB,E0:'               SET  3,B
1570 DATA 77:'          L2:     LD   (HL),A
1580 DATA 23:'                  INC  HL
1590 DATA 7E:'                  LD   A,(HL)
1600 DATA CB,07:'               RLC  A
1610 DATA 30,02:'               JR   NC,L1:
1620 DATA CB,E8:'               SET  2,B
1630 DATA 77:'          L1:     LD   (HL),A
1640 DATA 23:'                  INC  HL
1650 DATA 7E:'                  LD   A,(HL)
1660 DATA CB,07:'               RLC  A
1670 DATA 30,02:'               JR   NC,L0:
1680 DATA CB,F0:'               SET  1,B
1690 DATA 77:'          L0:     LD   (HL),A
1700 DATA 23:'                  INC  HL
1710 DATA 7E:'                  LD   A,(HL)
1720 DATA CB,07:'               RLC  A
1730 DATA 30,02:'               JR   NC,GRBYTE:
1740 DATA CB,F8:'               SET  0,B
1750 DATA 77:'          GRBYTE: LD   (HL),A
1760 DATA 78:'                  LD   A,B
1770 DATA CD,A5,00:'            CALL LPTOUT
1780 DATA ED,52:'               SBC  HL,DE
```

```
1790 DATA 0D:'                 DEC  C
1800 DATA 97:'                 SUB  A
1810 DATA B9:'                 CP   C
1820 DATA 20,A9:'              JR   NZ,KOLOM:
1830 DATA 19:'                 ADD  HL,DE
1840 DATA 23:'                 INC  HL
1850 DATA C1:'                 POP  BC
1860 DATA 10,A1:'              DJNZ TEKEN:
1870 DATA CD,9C,C0:'           CALL MOVELIN:
1880 DATA E1:'                 POP  HL
1890 DATA C1:'                 POP  BC
1900 DATA 10,83:'              DJNZ REGEL:
1910 DATA 3E,1B:'              LD   A,#1B
1920 DATA CD,A5,00:'           CALL LPTOUT
1930 DATA 3E,40:'              LD   A,#40
1940 DATA CD,A5,00:'           CALL LPTOUT
1950 DATA C9:'                 RET
1960 DATA 3E,0D:'     MOVELIN:LD   A,#0D
1970 DATA CD,A5,00:'           CALL LPTOUT
1980 DATA CD,A5,00:'           CALL LPTOUT
1990 DATA 3E,0A:'              LD   A,#0A
2000 DATA CD,A5,00:'           CALL LPTOUT
2010 DATA C9:'                 RET
2020 DATA 3E,1B:'     SETGR:   LD   A,#1B
2030 DATA CD,A5,00:'           CALL LPTOUT
2040 DATA 3E,47:'              LD   A,#47
2050 DATA CD,A5,00:'           CALL LPTOUT
2060 DATA 3E,30:'              LD   A,#30
2070 DATA CD,A5,00:'           CALL LPTOUT
2080 DATA 3E,36:'              LD   A,#36
2090 DATA CD,A5,00:'           CALL LPTOUT
2100 DATA 3E,30:'              LD   A,#30
2110 DATA CD,A5,00:'           CALL LPTOUT
2120 DATA 3E,30:'              LD   A,#30
2130 DATA CD,A5,00:'           CALL LPTOUT
2140 DATA 3E,32:'              LD   A,#32
2150 DATA CD,A5,00:'           CALL LPTOUT
2160 DATA 3E,35:'              LD   A,#35
2170 DATA CD,A5,00:'           CALL LPTOUT
2180 DATA 3E,36:'              LD   A,#36
2190 DATA CD,A5,00:'           CALL LPTOUT
2200 DATA C9:'                 RET
2210 REM              BUFFER:  EQU  $
```

```
1 '*********************************
2 '* HARDCOPY ROUTINE VAN SCREEN 2 *
3 '*  NAAR SEIKOSHA SP800 PRINTER   *
4 '*********************************
5 '
1000 CLEAR 200,&HBFFF
1010 FOR I=0 TO 190
1020 READ MC$:POKE &HC000+I,VAL("&H"+MC$)
1030 NEXT I
1040 DEFUSR0=&HC000
1050 KEY5,"GRPRI=USR0(0)"+CHR$(13)
1060 END
1070 DATA 3E,1B:'                LD   A,1B
1080 DATA CD,A5,00:'             CALL LPTOUT
1090 DATA 3E,41:'                LD   A,42
1100 DATA CD,A5,00:'             CALL LPTOUT
1110 DATA 3E,08:'                NOP
1120 DATA CD,A5,00:'             NOP
1130 DATA 21,00,00:'             LD   HL,0
1140 DATA 06,18:'                LD   B,24
1150 DATA C5:'          REGEL:   PUSH BC
1160 DATA CD,AA,C0:'             CALL SETGR:
1170 REM COPY LINE VRAM --> RAM
1180 DATA 11,BF,C0:'             LD   DE,BUFFER:
1190 DATA 01,00,01:'             LD   BC,#0100
1200 DATA CD,59,00:'             CALL LDIRMV
1210 DATA 11,00,01:'             LD   DE,#0100
1220 DATA 19:'                   ADD  HL,DE
1230 DATA E5:'                   PUSH HL
1240 DATA 21,BF,C0:'             LD   HL,BUFFER:
1250 DATA 06,20:'                LD   B,32
1260 DATA C5:'          TEKEN:   PUSH BC
1270 DATA 0E,08:'                LD   C,8
1280 DATA 11,07,00:' KOLOM:      LD   DE,7
1290 DATA 06,00:'                LD   B,0
1300 DATA 7E:'                   LD   A,(HL)
1310 DATA CB,07:'                RLC  A
1320 DATA 30,02:'                JR   NC,L6:
1330 DATA CB,F8:'                SET  0,B
1340 DATA 77:'          L6:      LD   (HL),A
1350 DATA 23:'                   INC  HL
1360 DATA 7E:'                   LD   A,(HL)
1370 DATA CB,07:'                RLC  A
1380 DATA 30,02:'                JR   NC,L5:
1390 DATA CB,F0:'                SET  6,B
1400 DATA 77:'          L5:      LD   (HL),A
```

MSX2 Zakboekje

In dit ene boek zijn zoveel mogelijk gegevens samengebracht, waar mogelijk in de vorm van overzichten en tabellen. Als u zelf al enige tijd bezig bent met het programmeren van uw MSX of MSX2 computer, dan zult u hebben ontdekt, dat u tijdens het programmeren een groot aantal boeken en tijdschriften moet naslaan om er de gewenste gegevens in te vinden.

Om u het zoeken in verschillende dokumenten te besparen, heeft de auteur alle tijdens het programmeren en gebruiken van de MSX en MSX2 computers benodigde gegevens zo overzichtelijk mogelijk samengebracht. In grote lijnen zult u de volgende gegevens in dit boek vinden:

- Algemene (konversie) tabellen
- Basic-tabellen, met onder meer een kompleet overzicht van alle Basic statements inklusief RS232C en MSX2 uitbreidingen
- Machinetaal tabellen
- Interface connectors
- BIOS entry points
- Disk I/O entry points
- Systeem geheugenadressen
- Voorbeeldprogramma's

Of u nu in Basic, in machinetaal, of in Basic met gebruikmaking van de aanwezige ROM-routines werkt; dit naslagboek zal u een hoop zoektijd besparen.

ISBN 90 6398 224 0 uitgeverij Stark-Texel